

AD-A041 042

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT--ETC F/G 1/3  
INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS.(U)  
1977

UNCLASSIFIED

AGARD-06GRAPH-224

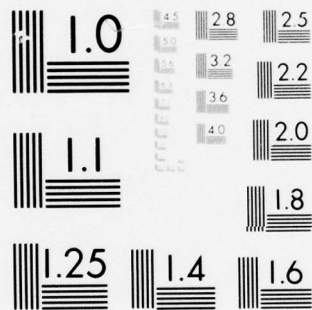
NL

1 OF 4

AD  
A041042







1  
B.S.

AGARD-AG-224

AGARD-AG-224

AD A 041 042

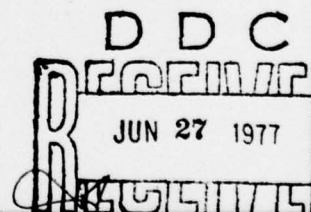
# AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT

7 RUE ANCELLE 92200 NEUILLY SUR SEINE FRANCE

AGARDograph No. 224

## Integrity in Electronic Flight Control Systems



NORTH ATLANTIC TREATY ORGANIZATION



DISTRIBUTION AND AVAILABILITY  
ON BACK COVER

DISTRIBUTION STATEMENT A

Approved for public release;  
Distribution Unlimited

UUC FILE COPY

NORTH ATLANTIC TREATY ORGANIZATION  
 ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT ✓  
 (ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARDograph No. 224

INTEGRITY IN ELECTRONIC  
 FLIGHT CONTROL SYSTEMS

Technical Director: Peter R. Kurzhals

ACCESSION FOR	
NTIS	White Section <input checked="" type="checkbox"/>
GOC	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

This AGARDograph was prepared at the request of the  
 Guidance and Control Panel of AGARD-NATO

DISTRIBUTION STATEMENT A  
 Approved for public release;  
 Distribution Unlimited

## THE MISSION OF AGARD

The mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Exchanging of scientific and technical information;
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Providing scientific and technical advice and assistance to the North Atlantic Military Committee in the field of aerospace research and development;
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Program and the Aerospace Applications Studies Program. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

The content of this publication has been reproduced directly from material supplied by AGARD or the authors.

Published April 1977

Copyright © AGARD 1977  
All Rights Reserved

ISBN 92-835-0192-6



*Printed by Technical Editing and Reproduction Ltd  
Harford House, 7-9 Charlotte St, London, W1P 1HD*

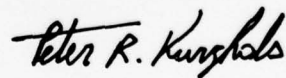
## PREFACE

With the growing dependance of aircraft designers on analog and digital flight control systems to provide increased operational capabilities and cost-effectiveness, development of design, implementation and test techniques which can assure the integrity of such systems has become critical to NATO mission success. This AGARDograph, assembled by the Guidance and Control Panel of AGARD, brings together related experience in the NATO community as a guide for future aircraft developments.

The intent of the AGARDograph is to address the hardware and software interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some gains in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.

The AGARDograph is organized into three major parts. Part I, *Background and Requirements*, includes overviews of the historical evolution of flight control systems and their reliability characteristics, as well as discussions of current and projected reliability trends for flight-critical control applications. Part II, *Analysis and Testing*, deals with theoretical, simulation and online techniques for failure detection and prediction. Principal areas addressed here are reliability modeling and analysis to achieve design integrity for complex systems, self-diagnosis and fault-identification methods, efficient software generation and management, preflight checkout and built-in test and evaluation. Part III, *Design and Implementation*, is devoted to representative high-integrity system mechanizations. Included are design approaches for redundant flight control sensors, processors, and actuators and selected implementations for fighter, transport and helicopter applications.

The participation and support of the many individuals who have made this AGARDograph possible is gratefully acknowledged. Particular recognition should be given to the extensive efforts of the authors in developing the overview and application papers and to their outstanding cooperation in sharing their experiences and views. The Editor also wishes to express his appreciation to Mr Colin Copage, UK, Dr Walter Metzdorf, Germany, Mr Alain Chadeau, France, and Mr Lawrence Taylor, USA, who served as coordinators for the identification and selection of topics and authors from their respective countries; to the Guidance and Control Panel members and AGARD staff for their valuable assistance and advice; and to Ms Nancy Owen who handled much of the correspondence and administrative support for the AGARDograph. The capable review of the AGARDograph by Mr H. Andrews, USA, contributed significantly to development of the final document.



Peter R. Kurzahls, Director  
Guidance, Control and  
Information Systems Division  
NASA Headquarters  
Washington, D.C.



## CONTENTS

	Page
<b>PREFACE</b> by P.R.Kurzhals	iii
	Reference
 <b><u>PART I – BACKGROUND AND REQUIREMENTS</u></b>	
<b>A HISTORICAL PERSPECTIVE FOR ADVANCES IN FLIGHT CONTROL SYSTEMS</b> by D.McRuer and D.Graham	1
<b>CHRONOLOGICAL OVERVIEW OF PAST AVIONIC FLIGHT CONTROL SYSTEM RELIABILITY IN MILITARY AND COMMERCIAL OPERATIONS</b> by S.S.Osder	2
<b>SAFETY CRITERIA FOR FAIL-OPERATIONAL AUTOLAND SYSTEMS AND THEIR APPLICATION</b> by D.Warren	3
<b>FUTURE TRENDS IN HIGHLY RELIABLE SYSTEMS</b> by J.Arnold	4
 <b><u>PART II – ANALYSIS AND TESTING</u></b>	
<b>A SURVEY OF DESIGN METHODS FOR FAILURE DETECTION IN DYNAMIC SYSTEMS</b> by A.S.Willsky	5
<b>CAST – A COMPLEMENTARY ANALYTIC-SIMULATIVE TECHNIQUE FOR MODELING COMPLEX, FAULT-TOLERANT COMPUTING SYSTEMS</b> by R.B.Conn, P.M.Merryman and K.L.Whitelaw	6
<b>SOFTWARE RELIABILITY: ANALYSIS AND PREDICTION</b> by M.L.Shooman	7
<b>SOFTWARE INTEGRITY THROUGH VISIBILITY</b> by G.Belcher and T.Egan	8
<b>TECHNIQUES FOR MICROPROGRAM VALIDATION</b> by W.C.Carter, H.A.Ellozy, W.H.Hoyner, Jr and G.B.Leeman, Jr	9
<b>SYSTEM INTEGRITY BY USE OF SELF DIAGNOSING FAILURE DETECTION</b> by R.Onken	10
<b>FAILURE SELF-DETECTION IN DIGITAL FLIGHT GUIDANCE SYSTEMS</b> by H.Drttil and W.Meyer	11
<b>SNEAK CIRCUIT ANALYSIS APPLICATION TO CONTROL SYSTEM DESIGN</b> by J.L.Wilson and R.C.Clardy	12
<b>BUILT-IN TEST TECHNIQUES FOR DIGITAL FLIGHT CONTROL SYSTEMS</b> by W.A.Plice	13
<b>PRE-FLIGHT DYNAMIC CHECKOUT</b> by D.R.Towill	14
 <b><u>PART III – DESIGN AND IMPLEMENTATION</u></b>	
<b>THRESHOLD REDUNDANCY MANAGEMENT WITH ARRAYS OF SKEWED INSTRUMENTS</b> by J.E.Potter and M.C.Suman	15

	Reference
<b>TIME-DIVISION MULTIPLEXED DATA BUS INTEGRATION TECHNIQUES</b> by E.C.Gangl	16
<b>HIGHLY RELIABLE MULTIPROCESSORS</b> by N.D.Murray, A.L.Hopkins and J.H.Wensley	17
<b>OBJECTIVES FOR THE DESIGN OF IMPROVED ACTUATION SYSTEMS</b> by B.H.Earley	18
<b>F-16 FLIGHT CONTROL SYSTEM DEVELOPMENT</b> by D.L.Carlton	19
<b>JA-37 DIGITAL AUTOMATIC FLIGHT CONTROL SYSTEM (DAFCS)</b> <b>SELF-TEST DEVELOPMENT</b> by D.G.Bailey and K.Folkesson	20
<b>DESIGN AND TEST EXPERIENCE WITH A TRIPLY REDUNDANT DIGITAL</b> <b>FLY-BY-WIRE CONTROL SYSTEM</b> by K.J.Szalai, P.G.Felleman, J.Gera and R.D.Glover	21
<b>L-1011 FLIGHT CONTROL SYSTEM</b> by J.A.Flapper and E.O.Thronsen	22
<b>LES COMMANDES DE VOL DE CONCORDE</b> par M.Bossard et R.Déqué	23
<b>A HIGH-RELIABILITY, HIGH INTEGRITY FLIGHT CONTROL SYSTEM</b> <b>FOR HELICOPTERS</b> by P.Robinson, J.Meadows and C.M.Copage	24



**PART I**

**BACKGROUND AND REQUIREMENTS**

## A HISTORICAL PERSPECTIVE FOR ADVANCES IN FLIGHT CONTROL SYSTEMS

Duane McRuer  
President and Technical Director, Systems Technology, Inc.  
Hawthorne, California 90250 U. S. A.

Dunstan Graham  
Technical Director, Systems Technology, Inc., and  
Professor, Princeton University  
Princeton, New Jersey 08540 U. S. A.

### INTRODUCTION

For the last quarter century, the automatic control of flight has been at the leading edge of aeronautical development. Automatic flight today retains this position at the technological leading edge while also exhibiting many features of maturity — one is that a great many people, large resources, and much time are needed to bring any new concept to fruition. That the fruit is worth growing can hardly be of doubt to any of us who rely on the modern airliner. This is routinely guided and controlled from climb-out after takeoff to rollout after touchdown by automatic devices operating with unparalleled reliability, extraordinary precision, and remarkable safety. It is even more evident in the military domain, where automatic flight is more often than not an absolute necessity.

Our intent in this paper is not to describe where we are in automatic flight, but rather to trace its development, so as to set the stage and to provide some perspective for what follows in this monograph. The evolution is interesting both as a tale of science and technology and as a case study of the rise of a major high technology industry. In both aspects the branches of the story came to a junction point shortly after World War II. Since that time there has been an enormous amount of activity over a very wide front which has been well documented. We will touch on this most modern era, but will emphasize the formative stages — those years which lead to the establishment of automatic flight control as a viable and essential discipline in aeronautics.

There is another highly practical reason for a history of this kind, as perhaps best expressed in the philosopher George Santayana's remark, "Those who do not remember the past are condemned to repeat it." It is in this vein as well that we will speak about the history of automatic flight, which has had its successes and also its failures, its dominant and subordinate trends sometimes recurring, and trace these successes and failures as a lesson for the future.

The theme for our history will encompass the early independent development of theory and practice, their subsequent confluence, and some relation of the perspective so gained to the present state of affairs.

### BEFORE FIRST FLIGHT — THE SEARCH FOR INHERENT STABILITY

Many of the fundamental problems of flight control were apparent before the Wright Brothers. For instance, the necessity for the control of the path of the airplane was recognized both practically and theoretically. An early example is the paths shown in Fig. 1, Lanchester's famous phugoids; on a larger scale the date, November 1897, can be seen in the lower left corner of this figure.

Lanchester was interested in the stability of the paths of airplanes, and he pursued this interest both on an experimental and theoretical basis. This was particularly easy to do with the path of gliders as they were launched. We suspect that most of the readers have at one time or another folded up a paper airplane and launched it. If the initial speed is just exactly right, the path is a straight line (ideally horizontal is drag is zero, as assumed by Lanchester). If the speed is higher, the glider will climb and may even loop, whereas if the launching speed is too slow it will sink, gain speed, and come up again. These paths are an expression of that fact. The horizontal line at the top is the path of an airplane launched at the correct speed. The other paths going through approximately sinusoidal flight and then those illustrating looping flight are the paths of airplanes that are launched with successively higher speeds.

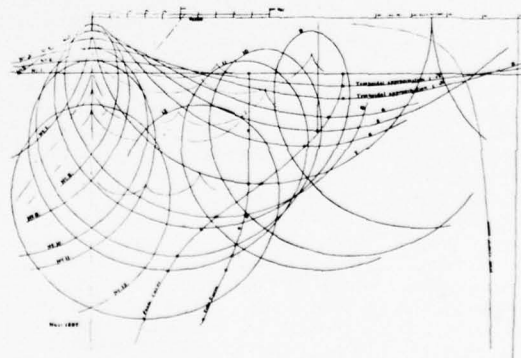


Figure 1. Lanchester's Phugoids

Lanchester published this chart in connection with a patent applied for in 1897. He was the first to study the stability of the path of the airplane with no control. He was in the tradition of the people who attempted to secure what they called "inherent" stability, that is, stability properties which accrued by virtue of fundamental structure — or, in modern terms, stability without feedback.

We now recognize that inherent stability is very unlikely to be achieved, but the early inventors pursued it with great diligence. Further, we appreciate today that the stability, if any, is largely with

respect to the air mass in which the airplane flies. The inherent stability which the early inventors attempted to secure by design of the aircraft itself is, in some cases, a major factor in present designs. For example, Pensaud, in his statement of the requirements for a horizontal tail, its angle of incidence, the location of the wing and tail with respect to the c.g., etc., is still followed in principle to achieve longitudinal stability. On the other hand, Lilienthal emphasized inherent lateral stability. He achieved it in several of his gliders, with the ultimate result that it killed him. The large effective dihedral of his hang glider created an inherent stability not with respect to the earth but with respect to the air mass, and on his last flight his control power was insufficient to offset a gust disturbance.

While the Wright Brothers are justly famed for their priority in many fields of aviation, their most notable contribution was the implicit appreciation that the secret to the control of flight was feedback. From their tethered and glider experiments they recognized that the human pilot, operating on feedback signals, that is, his attitude with respect to the ground, his position with respect to a desired landing point, etc., should be able to operate the controls so as to stabilize, control, and guide the aircraft in a desirable fashion. They recognized that the frustrating search for inherent stability might well be abandoned if only the pilot were provided with sufficiently powerful controls with which to balance and steer — in other words, that the human pilot, operating on feedback signals, could use the controls to stabilize a neutrally stable or an inherently unstable aircraft. The Wrights proceeded to build this aircraft configured for good control and were ultimately able to demonstrate its compliance with the first flying quality specification — a single sentence from Signal Corps Specification 486, "During this trial flight of one hour it must be steered in all directions without difficulty and at all times under perfect control and equilibrium." If only we could achieve such concise and relevant, yet subtle, language in our modern flying quality specifications!

So, by virtue of the Wright Brothers, we have feedback control and control-configured vehicles as fundamental features in aircraft. Let us turn now to the main theme where the feedback control is accomplished in an automatic way. To do this, the story will be divided into three time periods.

#### THE FIRST PERIOD: 1890-1934

The first period ranges from about 1890 to 1934. During this time, two types of engineers were active — the dynamics theoreticians and the tinkerer/inventors. They were both interested in path control, but their approaches and views kept them in essentially complete isolation from each other. The tinkerer/inventors were interested in stabilizing the path with control apparatus which served to control the aircraft so as to duplicate gyroscopic references. They were not particularly concerned with the actual dynamics of the aircraft except as these might incidentally interfere with this process. Consequently, they understood little about the aircraft dynamics except as a source of lags, and these lags were often second order to those of the primitive actuation equipment. For this reason, control at best was exerted on the kinematic and long-term motions of the aircraft and the aircraft dynamics had relatively small effect.

On the other hand, the aircraft stability and control dynamicists, the theorists, emphasized empirical and mathematical treatments of the dynamics of the uncontrolled aircraft. In spite of the practical demonstration of the Wright Brothers, they were after inherent stability and they studied the subject for that purpose. The most perceptive of the theorists believed from the start that the "controlled aircraft" was the important thing, but that pilot actions were too complex and variable to analyze. "Automatic stability" depending on the use of gyros, pendulums, or other movable parts to create airplane stability was not well thought of. For example, the great pioneer of aircraft stability and control, G. H. Bryan, said in 1910, while speaking of the stability of aircraft (Ref. 1):

"Apart from the fact that movable parts are liable to get out of order, it must be remembered that they increase the number of degrees of freedom of the machine, thus further adding to the number of conditions which have to be satisfied for stability — a number quite large enough already. I anticipate that the successful aeroplane of the future will possess inherent, not "automatic" stability, movable parts being used only for purposes of steering."

The first among the tinkerer/inventors was Sir Hiram Maxim, an expatriate Yankee, whose life provided one of Don Ameche's many biographical movie roles. Maxim was a prodigious inventor, most famous perhaps, and certainly rich because of, the Maxim machine gun. In 1891 he began the construction of a giant flying machine (Fig. 2). Photographs are available, but they don't convey the impression that this drawing does of this fabulous and colossal machine. It was 110 ft long and weighed 3-1/2 tons. It was powered with a steam engine. Maxim configured his experiments in such a way that the machine had about 6 inches between its support on a rail and a "tether rail" which was designed to restrain it from really flying. In one of his very early trials the machine actually lifted enough to break the restraining tether rail on one side, whereupon it rolled over and crashed. Maxim, who by his own account intended to secure his fame in aeronautics with discussions of lift and power, thought that the point had been proved and abandoned his aeronautical experiments after an expenditure of nearly £20,000.

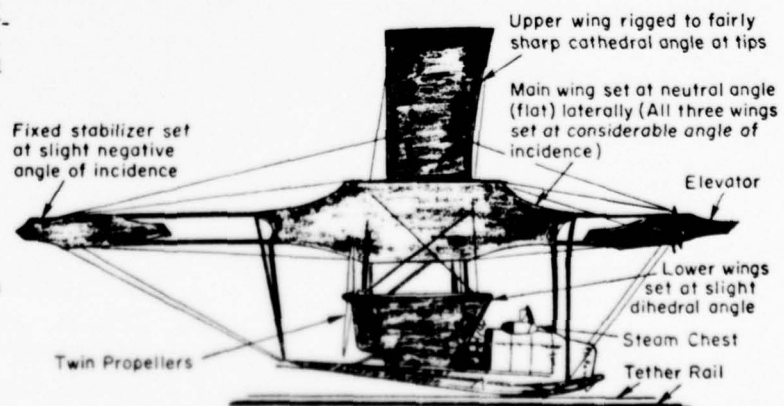


Figure 2. Maxim's Flying Machine



Now, why do we show you this? It is because Maxim, the tinkerer/inventor, had installed in this giant flying machine what we would now call a stability and control augmentation system of an incredibly advanced design for the time. Figure 3 is a drawing, from Ref. 2, of this apparatus. In his book, Artificial and Natural Flight (Ref. 3), Maxim described his pendulous gyroscope as steam driven "after the fashion of a Barker's mill." A Barker's mill is a mechanical contrivance invented by a Dr. Barker about the end of the seventeenth century. It consists of a hollow cylinder provided with a number of horizontal arms fitted with lateral apertures, mounted so as to rotate about the cylinder's axis of symmetry. In Maxim's case, steam provided the motive force. This was Maxim's gyroscope drive principle. The gyroscope was connected to a valve, and the valve body is repositioned by the output of a servo. It is interesting to note that the output could also be positioned by an "engineer" with his pitch command lever. So, we have attitude hold and provision for pilot intervention with a high-performance servo system and a gravity-erected gyroscope. The date — approximately 1894. The airplane did not fly, but if it had, the autopilot might well have worked.

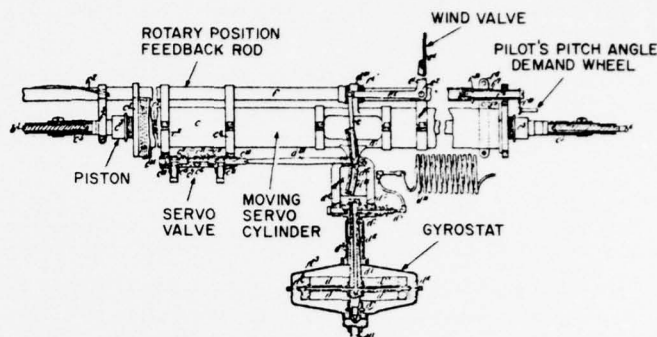


Figure 2. Maxim Stabilizer

Somewhat later but still very early, Elmer Sperry began experimenting with gyroscope stabilization of the airplane. After some abortive first efforts in 1909-10, in the period from 1910 to 1914 he and his son Lawrence, who was keenly interested in aeronautics, developed what they called a "stabilizer" and which was in fact an autopilot.

Most readers of this paper have probably seen the photograph, which we will not show, of the demonstration of the Sperry stabilizer in 1914 winning a safety prize donated by the Aero Club of France. The way in which they demonstrated the pitch and roll control was for Lawrence Sperry to fly at low altitude over the Seine standing up in the cockpit of his Curtiss Flying Boat holding his hands over his head while his mechanic walked out on the wings. In this way they exhibited the capabilities of the control system and, simultaneously and incidentally, a most touching faith in the reliability of the equipment! Figure 4 illustrates the equipment out of the Flying Boat. The heart of the system was a gyro platform — four gyroscopes on a pendulously suspended gimballed platform. The gyros maintained the platform alignment. Their spin axes were all horizontal, with one pair aligned longitudinally and the other pair transversely. The gyro pairs had oppositely spinning wheels and were coupled for equal and opposite precession. Roller contacts on the stabilized platform measuring the bank and pitch angles actuated solenoid clutches in pneumatic servos connected to the ailerons and elevator; motion of the surfaces repositioned the contactor elements. An engine-driven alternator provided gyro power and the servos were powered by compressed air from an engine cylinder. It worked very well, but noisily.

Other inventors were very busy. Starting about the time that flying came to Europe, people tried or conceived of all kinds of automatic stabilization for an aircraft. They used the feedback of speed, of incidence (what we now call angle of attack), of inclination (what we now call pitch angle), of its derivative, etc., and they attempted power amplification and servo mechanism drives of the control surfaces. Figure 5, which is not complete, is in fact taken from a 1936 NACA TM (Ref. 4). The chart is simply an abstraction of a vast amount of work. Perhaps a sad part of all this vast experimentation on feedback control of aircraft was that nobody had any use for it. The designers of aircraft had learned how to provide enough stability so that the pilots could handle the airplanes and nobody needed feedback control.

In 1932 a three-axis Sperry autopilot, very similar to the original 1912-14 system with the addition of a rudder axis, was bought by Eastern Airlines for their Curtiss Condors. The Condor was the

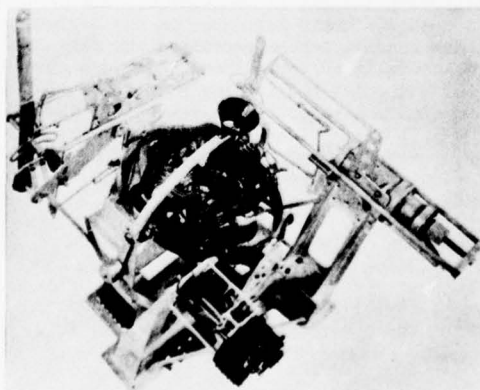


Figure 4. Sperry Airplane Stabilizer in 1914

FEEDBACK VARIABLE	CONTROL	INVENTOR AND DATE	ACTUATING MEANS
Speed, $U$	Elevator Deflection, $\delta_e$	Budig 1912 Etevé 1914	Mechanical connection to sensor
"Incidence", $\alpha$	" " "	Etevé 1910	" " "
"Inclination", $\theta$	" " "	Regnard 1910 Sperry 1912 R. A. E. 1927	Electric type of servo Air-turbine-driven clutch servo Pneumatic servo
Angular Velocity, $\dot{\theta}$	" " "	Girardville 1910	Mechanical connection to sensor
Direction of apparent gravity, $g \sin \theta + dU/dt$	" " "	Moreau 1912	Electric-motor-driven clutch servo
Speed, $U$ , and "Inclination", $\theta$	" " "	Marmontier 1909	Unknown type of servo
Bank Angle, $\phi$	Aileron Deflection, $\delta_a$	Sperry 1912	Air-turbine-driven clutch servo
Heading, $\psi$	Rudder Deflection, $\delta_r$	R. A. E. 1927	Pneumatic servo
Yawing Velocity, $r$ , and Side Acceleration, $a_y$	" " "	Mozade and Aveline 1922	Pneumatic servos

Figure 5. Early Flight Control Inventions

last of the biplane transports in service in the United States, and its autopilot was the last of the primitive devices. By 1933 the Sperry Gyroscope Co. had developed their A-2 autopilot. The A-2 was much more modern in concept and execution. It was essentially proportional attitude control. It had pneumatic pickoffs on the gyros and hydraulic servos. Its first practical application was to the around-the-world flight of Wiley Post between the 15th and 22nd of July in 1933, when he flew his Lockheed Vega 5B, the "Winnie Mae," around the world in a total flying time of 115 hours, 36-1/2 minutes. Post gave the A-2 autopilot, "Mechanical Mike," credit for helping the success of this incredible flight. He was able to nap in flight while the aircraft was under automatic flight, again a most touching faith in the reliability of the autopilot! The New York Times called the flight "a revelation of the new art of flying." The news report added:

"By winning a victory with the use of gyrostats...Post definitely ushers in a new stage of long-distance aviation. The days when human skill alone, an almost birdlike sense of direction, enabled a flyer to hold his course for long hours through a starless night or over a fog are over. Commercial flying in the future will be automatic."

By 1934 this same autopilot, the Sperry A-2, was ordered and installed in what is now United Airlines' Boeing 247s. Just as the Condor was the last of the biplane fabric-wing transports, the Boeing 247 was the first of the all-metal monoplanes. Thus, 1934 provides an appropriate end for the first era of automatic control of aircraft from the standpoint of the tinkerer/inventor.

In the meantime, the theoreticians had not been idle. Lanchester and his study of the phugoid motions has already been mentioned. Not very long after Lanchester's investigations, in fact in 1903, the year of the first flight, we have the first major contribution of Bryan. For starters, he studied the linearized motions of the airplane, assuming small perturbations; discovered the separation of the longitudinal and lateral motions; invented stability derivatives, etc. Only the orientation of his axis system differed from modern usage! Shortly after that, Bairstow and Melvill Jones, at the National Physical Laboratory in Great Britain, measured the stability derivatives and calculated the motions of practical airplanes (e.g., Ref. 5). In the period from about 1910 through the early 30's there was an enormously productive effort in Great Britain. People calculated the stability of aircraft, calculated the response to disturbances, calculated the response to applications of controls, made full-scale in-flight measurements to show that the responses were correct, etc.

Perhaps most notable from the automatic control standpoint during this period are the efforts of Gates and Garner. Gates, in 1924 (Ref. 6), assumed that the controls were moved according to certain "laws," that is, in proportion to certain output variables and their derivatives. He stressed that good stability was not enough, that it was essential also to consider the amplitudes of the several modes of motion. In 1926 Garner (Ref. 7) made an analysis of the lateral-directional motions of an airplane under the influence of feedback control. He specifically pointed out that the movements of the controls might be regarded as made either by the human pilot or by some mechanical means. Garner further had the wit and vision to make provision in the theoretical treatment for lag in the application of controls and was able to point to a qualitative correspondence between his analytical results and flight tests of an RAE automatic rudder control which had an appreciable lag.

It now seems surprising that these papers are not given more prominence in accounts of the development of the theory of automatic control systems. They seem, in fact, to have fallen in a deep dark hole. Perhaps they were simply too far ahead of their time; perhaps, on the other hand, it was only in Great Britain where automatic flight control system development at this time was the responsibility of a government research establishment that it was thought to be desirable to make response calculations in connection with the design of "practical" systems. It cannot be said that the people who were developing autopilots paid no attention to the theoreticians; they were sitting across the hall from one another and they did know what the theoreticians were doing. For example, as early as 1937 in the dawn of the second era we have the paper by Meredith and Cooke (Ref. 8). They crossed the lines by describing both the practical and theoretical aspects of autopilot development.

By 1935 when B. Melvill Jones surveyed stability and control in "Dynamics of the Airplane," Section N of Vol. V of Durand's Aerodynamic Theory (Ref. 9), the classical approach initiated by Bryan was well-established but very little used. The theory of small perturbations, the examination of stability, the ability to calculate the time history in response to disturbance or to the application of control, the full-scale experiments that led to the conviction that the theory of infinitesimal motions was practical for the prediction of stability of motion, etc., were all meticulously and elegantly covered. The effects of variations in the configuration of a typical airplane were traced via their influence on the derivatives to the result in terms of stability in motion. Furthermore, these results were appreciated not only in terms of the solutions to specific numerical examples, but more generally as approximate solutions given in terms of the dominant literal stability derivatives. But, Melvill Jones did not cover feedback control of the aircraft's motions although he wrote a decade after Gates' initial efforts. He recognized that:

"It is probable that mechanical control will become increasingly popular for large long-distance aeroplanes and for anything in the nature of pioneer work in this subject calculations of this kind are essential. No mention of the methods of extending the calculations to deal with mechanical control will, however, be found in the present work since this is still a matter of research and what little has been published is mainly of a controversial nature."

He did recognize that "work of the type discussed here forms an essential introduction to the study of mechanical control." Melvill Jones' comment on the application of the theory which he did cover, i.e., aircraft-alone dynamics, was:

"In spite...of the completeness of the experimental and theoretical structure... it is undoubtedly true that, at the time of writing, calculations of this kind are very little used by any but a few research workers. It is in fact rare for anyone actually engaged upon the design and construction of aeroplanes to make direct use

of [such] computations..., or even to be familiar with the methods by which they are made....In my own opinion it is the difficulty of computation...which has prevented designers of aeroplanes from making use of the methods...."

We shall refer again to this quotation. But it does, by extension, make matters clear about automatic flight as well. Since the procedures then available for treating automatic systems involved factoring quintics or higher degree polynomials, whereas the aircraft-alone equations were only quartics, it is easy to see why very few people were interested in pursuing design calculations in any depth.

#### THE SECOND PERIOD: 1934-1947

Let us turn now to the second era which persisted from 1934 to 1947. The approaching Second Great War dramatically influenced the development of automatic pilots and encouraged the further elaboration of the theory, but they still remained largely separate lines of endeavor. What happened, in the United States anyway, was the very rapid development of what was then called the all-electric automatic pilot. Recall that the Sperry 1914 autopilot was electric in its sensors and pickoffs but not in its actuation. Subsequently, the Sperry Co. went to pneumatic pickoffs, pneumatic power for the gyroscopes themselves, and hydraulic actuation. The all-electric autopilots were in fact all-electric in the sensors, pickoffs, power amplification, and actuation (Ref. 10). The flexibility associated with this means of mechanization permitted rapid introduction of a number of novel features — a single-knob turn control (replacing three different knobs), erection cutout, altitude and heading as outer loops superimposed around the previous pitch and bank loops, synchronizers, rate gyros or electrical compensation to increase damping — all appeared in practical production flight hardware within a very short time. Again, almost all of this was accomplished by the tinkerer/inventors operating with little or no theoretical backup. Like aircraft themselves, the stability and control properties of the closed-loop systems were evaluated in flight tests, and flight control equipment was also designed with the aid of extensive full-scale testing. The "curse of dimensionality" (with apologies to Richard Bellman) mentioned by Melvill Jones was still present, and cut-and-try did the job — indeed, so well that all the elements of a modern automatic pilot were now at hand.

The triumph of the tinkerer/inventors came in 1947. On the second column of the page from the New York Times for September 23, 1947, shown in Fig. 6 is an article which describes the flight of the U. S. Air Force's All-Weather Flying Division's C-54, "Robert E. Lee." This aircraft had a Sperry A-12 autopilot with approach coupler and a Bendix automatic throttle control. These were more or less state of the art at the time. It also had some fairly special-purpose IBM equipment which permitted commands to its automatic control to be stored on punched cards fed automatically. From the time that the brakes were released for takeoff from Stephenville, Newfoundland, until the landing roll was completed at Brize-Norton in England the next day, no human hand touched the control. The selection of course, radio station, speed, flap setting, landing gear position, and the final application of wheel brakes were all accomplished automatically from a program stored on punched cards. The complete automation of aircraft flight appeared again to be at hand.



Figure 6.

The second era also saw the very rapid development of theory with which we are familiar today. Servo analysis techniques as they derived from feedback amplifier design were introduced first to servo-mechanisms and later to aircraft. The key contributions of Nyquist (Ref. 11), Bode (Ref. 12), Nichols and Phillips (Ref. 13), Harris (Ref. 14), Hall (Ref. 15), the stability diagrams (now called parameter spaces), Evans' root locus (Ref. 16), time vectors (Refs. 17-18), etc., were all developed during this era. Although they were scarcely ever applied to automatic flight control system design, the techniques were there waiting in the wings — theories in search of problems.

#### THE MODERN PERIOD: 1947-

The problems were not long in coming. The war had seen the advent, on both sides, of the turbo-jet engine, and suddenly the limits of the flight envelope were enormously extended in both speed and altitude, with concomitant configuration changes involving increased wing loadings, mass distributions concentrated in long thin fuselages, the aerodynamic benefits of short span, swept wings, etc. All sorts of new problems arose that were of interest both to the aircraft designer and to his new fixit man, the flight control designer. New phenomena were even discovered — fuel slosh, rolling instability, structural instabilities influenced by automatic control, etc. Power-booster controls came into use to handle the large hinge moments of the control surfaces, and these actuators had stability difficulties of their own. All of these trends were bad news for the automatic flight control system designer, who now desperately needed and wanted analytical help. People suddenly seemed to realize that pooling the knowledge of dynamic stability with the knowledge of instrument design was essential for the betterment of aeronautics if this was to be accomplished in an expeditious way without expenditure of an excessive number of experimental flight hours each fraught with extraordinary adventures for test pilots! So, while the intimate joining of control technology and vehicle dynamic analysis would no doubt have come about in any event, it was forced by the marked deficiencies in stability of the new jet aircraft and by the advent of the guided missile,



where it was obviously essential to match the dynamics of the airframe and the control system from the first flight on. This is the confluence of theory and practice. One of us likes to date this as 1947 to 1948 and associate it, admittedly on a personal basis, with a remarkable airplane now little remembered.

Figure 7 shows the B-49, which in 1948 was to be the production bomber for the United States Air Force. It was the last and most successful of John Northrop's great series of all-wing aircraft. In our modern jargon, it was a control-configured vehicle, and its great success as a flying machine was peculiarly dependent upon many flight control system developments. Its control surfaces were powered by the first successful fully-powered hydraulic actuators developed in the United States on its propeller-driven version, the XB-35, first flown in 1946. These were essential because of anticipated (and actual) unstable hinge moment gradients due to increasing separation over the trailing edge as stall was approached. The airplane was also equipped with a series-installed yaw damper/rudder trim system in a quasi-fly-by-wire configuration which was, as far as we know, the first successful stability augmentor flown in the United States. [K. H. Doetsch had earlier applied a bang-bang yaw damper to a rudder tab on the Henschel Hs 129 (Ref. 2)]. In fact, the very name "stability augmentor" stems from this aircraft. It was originally "stability derivative augmentor," but the middle word was deleted to more readily fit the title block of an installation drawing! Besides the obvious configuration aspects to maximize performance while attending to the consequent control problems via automatic control, considerable thought was given to further improvement of the landing and cruise performance by flying the aircraft with an unstable c.g. location. Analytical and experimental studies, including a flight demonstration, of stabilization of a 10 percent unstable aircraft with automatic control were undertaken and seriously considered for application. This was not adopted because the aircraft met requirements readily without the additional automatic system complexity. But the important thing for our story is that this is one of the first, if not the first, examples of the marriage of the science of the theoretician with the art of the tinkerer/inventor. Frankly, our recollection is that no one thought much about it at the time. The problems were foremost and their solution required the wedding even if a shotgun were needed. So, in short order, there was invented, or re-invented, in aircraft plants and autopilot companies all over the world, the yaw damper, pitch damper, roll damper, sideslip stability augmentor, transonic trim shifter, autothrottle, and other devices. These were applied with close connections between theory and practice to the alleviation of the new dynamical effects.



Figure 7. The B-49, Northrop Flying Wing

Unfortunately, in spite of the flight of the "Robert E. Lee" and the 2-1/2 decades since that time wherein theory and practice have been well connected, all of our problems in automatic flight are yet to be solved. The reason we continue to have problems and the reason why some of us as engineers are still employed in this business is that our hardware/software capabilities have expanded enormously and that the requirements are changeable and multifaceted and, often, somewhat subtle to appreciate. While the theoretical structure is well developed and practically applied in design, the actual selection of a design for a particular aircraft depends on a very large number of things which do not readily lend themselves to inclusion in, for example, a cost functional. The proper specification and satisfaction of all these desirable characteristics in the dawning new fourth era of automatic flight control will be central, for in this era the automatic control will be necessary for the successful performance of some aircraft in a majority, if not all, of the flight regimes. We are faced with major new challenges in which full-time, total-flight-envelope, flight control promises new dimensions of both aircraft and total system performance. The shibboleths of the new flight control technology are words like multimode, full-flight envelope, decoupled (roll/yaw, speed/flight path, rotation/translation), direct lift and direct side force, redundancy, graceful degradation, and other good words adopted by the autopilot salesman to describe the virtues of his products. To satisfy the interacting requirements and make good on the descriptive phrases requires the same kind of engineer for the fourth era as was developed and operated in the third. The details of the hardware and software for highly redundant and complex equipment at the fringe of the state of a particular hardware art can never be permitted to get too far from the comprehension of an analyst charged with overall system cognizance. At the same time, the vision of flight control theoreticians should never become so opaque as to provide results of theoretical interest only. The dangers of a new separation between theory and practice are, we believe, increasing. For example, as Melvill Jones noted, two generations ago the intellectual mathematical equipment of skilled stability and flight control system analysts generally exceeded their physical ability to perform the calculations which might be needed or desired. Nowadays, quite the opposite situation exists, because advances in both analog and digital computation allow the consideration of problems which at one time would have been rejected as being too time consuming. As a consequence, the analysts' physical means now often exceed his mental grasp, and what he can compute may far exceed his understanding or appreciation. This can lead to an excessively empirical approach to design which is similar to the one used by the tinkerers thirty or more years ago. But a key difference exists in abstractions involved. Regardless of the detail and complexity of our mathematical models, they remain just that, whereas the physical equipment and the aircraft which are the objects of our abstractions were the tinkerer's models. Viewed in these terms, too great a reliance on a numerical empirical approach to design is no better, and may be even worse, than the physical empiricism of earlier days. When inundated by computer printouts and strip chart recordings we are confronted with a crucial problem — what is the essence?



what does it all mean? And even when this is unraveled, paper studies are obviously only as good as the implicit underlying assumptions. No matter how prescient the engineer may be in analytical forecast of system normal and abnormal behavior, one invariably finds a reservoir of residual problems when the apparatus is built. Thus, in the fourth era of flight control, it is essential that we keep the tinkerer/inventor and the theoretician communicating so that the science of automatic flight control retains its maturity without progressing to senility.

#### REFERENCES

1. Bryan, G. H., Stability in Aviation, London, Macmillan and Co., 1911.
2. Howard, R. W., "Automatic Flight Controls in Fixed Wing Aircraft; The First 100 Years," Aeronautical Journal, Vol. 77, No. 755, Nov. 1973, pp. 533-562.
3. Maxim, H. S., Artificial and Natural Flight, London, Whittaker and Co., 1908, pp. 92-94.
4. Haus, Fr., Automatic Stabilization, NACA TM-802, Jan. and Feb. 1936.
5. Bairstow, L., B. Melvill Jones, and B. A. Thompson, Investigation Into the Stability of an Airplane, ARC R&M 77, 1913.
6. Gates, S. B., Notes on the Aerodynamics of Automatic Directional Control, RAE Rept. No. BA 487, 19 Feb. 1924, and Notes on the Aerodynamics of an Altitude Elevator Control, RAE Rept. No. BA 494, 19 Mar. 1924.
7. Garner, H. M., Lateral Stability with Special Reference to Controlled Motion, ARC R&M 1077, Oct. 1926.
8. Meredith, F. W., and P. A. Cooke, "Aeroplane Stability and the Automatic Pilot," J. Roy. Aeron. Soc., Vol. 61, No. 318, June 1937, pp. 415-436.
9. Melvill Jones, B., "Dynamics of the Aeroplane," in W. F. Durand, ed., Aerodynamic Theory, Vol. V, Pasadena, Calif., Durand Reprinting Committee, 1943, republished in New York, Dover Publications, 1963, pp. 2-3, 169.
10. Gille, W. H., and R. J. Kutzler, "Application of Electronics to Aircraft Flight Control," Trans. AIEE, Vol. 63, No. 1944, pp. 849-853.
11. Nyquist, H., "Regeneration Theory," Bell Systems Tech. J., Vol. 11, No. 1, Jan. 1932, pp. 126-147; see also "The Regeneration Theory," Trans. ASME, Vol. 76, No. 8, Nov. 1954, p. 1151.
12. Bode, H. W., "Relations Between Attenuation and Phase in Feedback Amplifier Design," Bell System Tech. J., Vol. 19, No. 3, July 1940, pp. 421-454; Network Analysis and Feedback Amplifier Design, New York, Van Nostrand Co., 1945.
13. James, H. M., N. B. Nichols, and R. S. Phillips, Theory of Servomechanisms, New York, McGraw-Hill Book Co., 1947.
14. Harris, H., Jr., The Analysis and Design of Servomechanisms, OSRD NDRC (Sec. D-2), Rept. 464, Jan. 1942.
15. Hall, A. C., The Analysis and Synthesis of Linear Servomechanisms, Cambridge, Mass., Technology Press, 1943.
16. Evans, W. R., "Graphical Analysis of Control Systems," Trans. AIEE, Vol. 67, 1948, pp. 547-551; "Control System Synthesis by the Root Locus Method," Trans. AIEE, Vol. 69, 1950, pp. 66-69.
17. Mueller, R. K., "A Graphical Solution of Stability Problems," J. Aeron. Sc., June 1937.
18. Doetsch, K. H., The Time Vector Method for Stability Investigations, ARC R&M 2945, 1957.

# CHRONOLOGICAL OVERVIEW OF PAST AVIONIC FLIGHT CONTROL SYSTEM RELIABILITY IN MILITARY AND COMMERCIAL OPERATIONS

S. S. Osder  
Sperry Flight Systems  
Phoenix, Arizona 85036  
USA

## SUMMARY

Two decades of flight control system mechanization advances are traced from the perspective of reliability. Despite dramatic advances in device technology and miniaturization the demand for more functions tended to exceed the progress in electronics. By the latter 1960's, complexity growth related to system monitoring and redundancy management reached limitations of analog technology and set the stage for introduction of digital flight control systems.

## 1.0 INTRODUCTION

The rapid advances in solid state electronic technology have made a very significant impact in avionics equipment. In many instances, the new electronics have provided the potential for a weight improvement of nearly two orders of magnitude over the state of the art in the 1950's. Automatic flight control systems have exploited this electronic progress to the greatest extent possible, and yet the newest systems are not significantly reduced in size over their predecessors of a decade ago. Regardless of the improvements in electronic miniaturization, they are always outstripped by demands for more functions and additional automation. We can indeed build the automatic flight control system of the 1950's in a fraction of its original size and with major improvements in reliability. However, there are no customers for this type of progress. The demand is for new functions that require automatic flight controls to play a far more important role in normal flight missions.

The technology transitioned from vacuum tube circuits to the first transistorized mimics of these tube circuits and finally to the large scale integrated devices that dominate the designs of the mid-1970's. The applications grew from the simple pilot relief autopilot to the redundant, flight-critical guidance and control systems that provided such functions as automatic landing, stabilization of marginally controllable air frames and fly-by-wire. The intent of this paper is to provide a perspective of how we evolved to the present state of the art in automatic flight control with emphasis on how the pursuit of reliability always challenged and motivated the technology advances.

## 2.0 REVIEW OF ELECTRONIC PROGRESS IN AUTOMATIC FLIGHT CONTROLS

Starting the review of automatic flight control technology with the first silicon autopilots, circa 1958, it can be shown that the electronic features of that era became liabilities within a few years. Table 1 illustrates the evolution of equipment features through six generations of design innovations that have occurred. The 1958 systems used electromechanical synchronizer/integrator computational modules, building-block circuits in the form of amplifiers, demodulators and modulators, miniature sealed relays to implement the various interlock functions, and external gain control potentiometers inserted in the control signal path. By 1960 to 1962, improved transistor performance and miniaturization resulted in techniques that obsoleted many of these 1958 features. Device miniaturization and reliability improvements allowed the use of embedded functional modules that enhanced maintainability. Solid-state switching began to replace the miniature relays despite the continued improvement and miniaturization of relays. Crude electronic multipliers eliminated the need to interrupt the control signal path with remote gain control potentiometers.

TABLE 1  
SIX GENERATIONS OF PROGRESS IN AUTOMATIC FLIGHT CONTROL ELECTRONICS

Generation	Time Period	Main Improvement	Equipment Characteristics
1	1958	First Silicon Transistor Autopilots	<ul style="list-style-type: none"> <li>• Electromechanical Instrument Servo Computing Elements</li> <li>• Building Block Circuits</li> <li>• Gain Control Potentiometers</li> <li>• Miniature Sealed Relays</li> </ul>
2	1960-1962	Planar Transistors - Miniature Embedded Circuits	<ul style="list-style-type: none"> <li>• Electronic Multipliers Replace Gain Control Potentiometers</li> <li>• Limited Introduction of Solid-State Switching to Replace Relays</li> <li>• Miniature Embedded Building Block Circuit Modules</li> </ul>
3	1964	DC Computation - Limited Microelectronics	<ul style="list-style-type: none"> <li>• Greatly Improved Computation Accuracy</li> <li>• Introduction of Solid-State Synchronizers Replace Electromechanical Computing Elements</li> </ul>

TABLE 1 (cont)  
SIX GENERATIONS OF PROGRESS IN AUTOMATIC FLIGHT CONTROL ELECTRONICS

Generation	Time Period	Main Improvement	Equipment Characteristics
4	1966-1968	All Microelectronic Autopilots	<ul style="list-style-type: none"> <li>• Microelectronic Subassembly Functional Modules</li> <li>• All Solid-State Switching</li> </ul>
5	1969-1974	MSI - LSI Components	<ul style="list-style-type: none"> <li>• Built-In Digital Data Interfaces</li> <li>• Multi-Function Monolithic and Hybrid Devices</li> </ul>
6	1974-	Digital Autopilots	<ul style="list-style-type: none"> <li>• General Purpose Digital Computer provides all Control and Logic</li> </ul>

By 1963 to 1965, improved transistor stability allowed the application of dc computation techniques to usher in an era of additional miniaturization resulting from the elimination of suppressed carrier data handling circuitry. Also, accuracy improvements as great as 10:1 over previous mechanizations were achieved. Almost all relays could now be eliminated, solid-state synchronizers replaced the electromechanical units, and a limited number of microelectronic circuits were introduced. By 1966 to 1968, the era of the all-microelectronic autopilot had arrived. A single microelectronic subassembly module 10 cubic centimeters in volume could perform complete signal processing and control law computation functions with self-contained digital logic and gain control interfaces. An overview of this evolution is summarized in Figure 1. Because of a change in the electronic techniques available, the system organization moved toward the use of miniature, self-sufficient subassembly computing and signal processing modules. An example of how this simplified system organization works is shown in Figures 2a and 2b. The earlier systems required complicated routing of control signals in and out of the computers for gain controls applied by external devices (Figure 2a). Indeed, some of the most unreliable designs appeared in the late 1950's, ironically as a result of improved frequency domain system analysis and synthesis techniques based on root locus S plane design criteria. The designers attempted to program control laws as a function of altitude, Mach number, speed and aircraft configuration so that the closed loop system poles always remained in narrow regions of the S plane. Subsequent insights recognized that an aircraft's disturbance and command responses could be significantly different so that relatively high gain control augmentation systems could be designed with very non-critical requirements for gain programming. The unwieldy complexity of the earlier systems, however, with their complex arrays of air data controlled potentiometers were a prime motivation for the enchantment with "adaptive control" as a solution to these unreliability problems in the late 1950's and early 1960's (Reference 1).

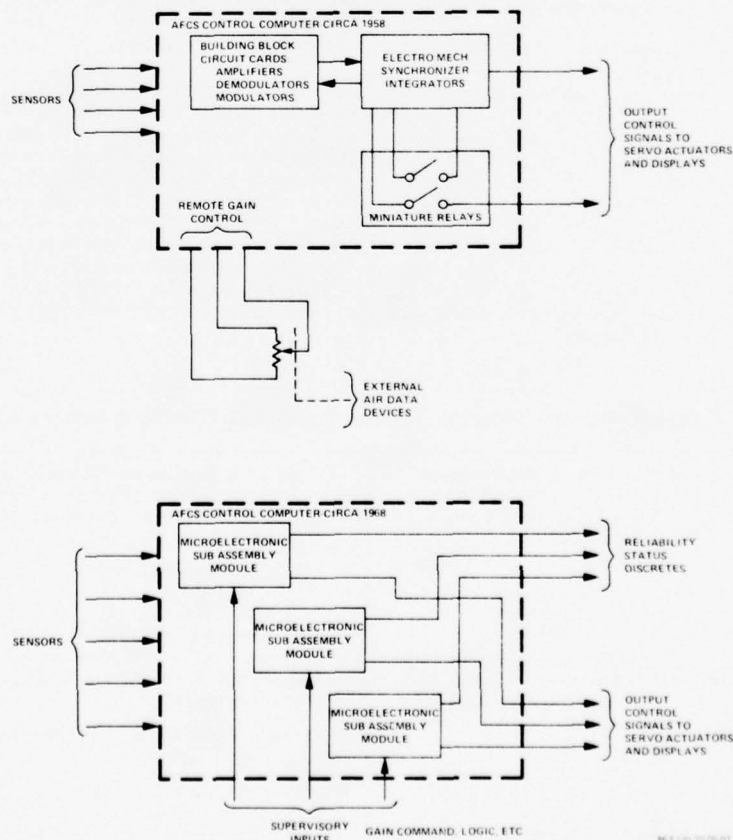
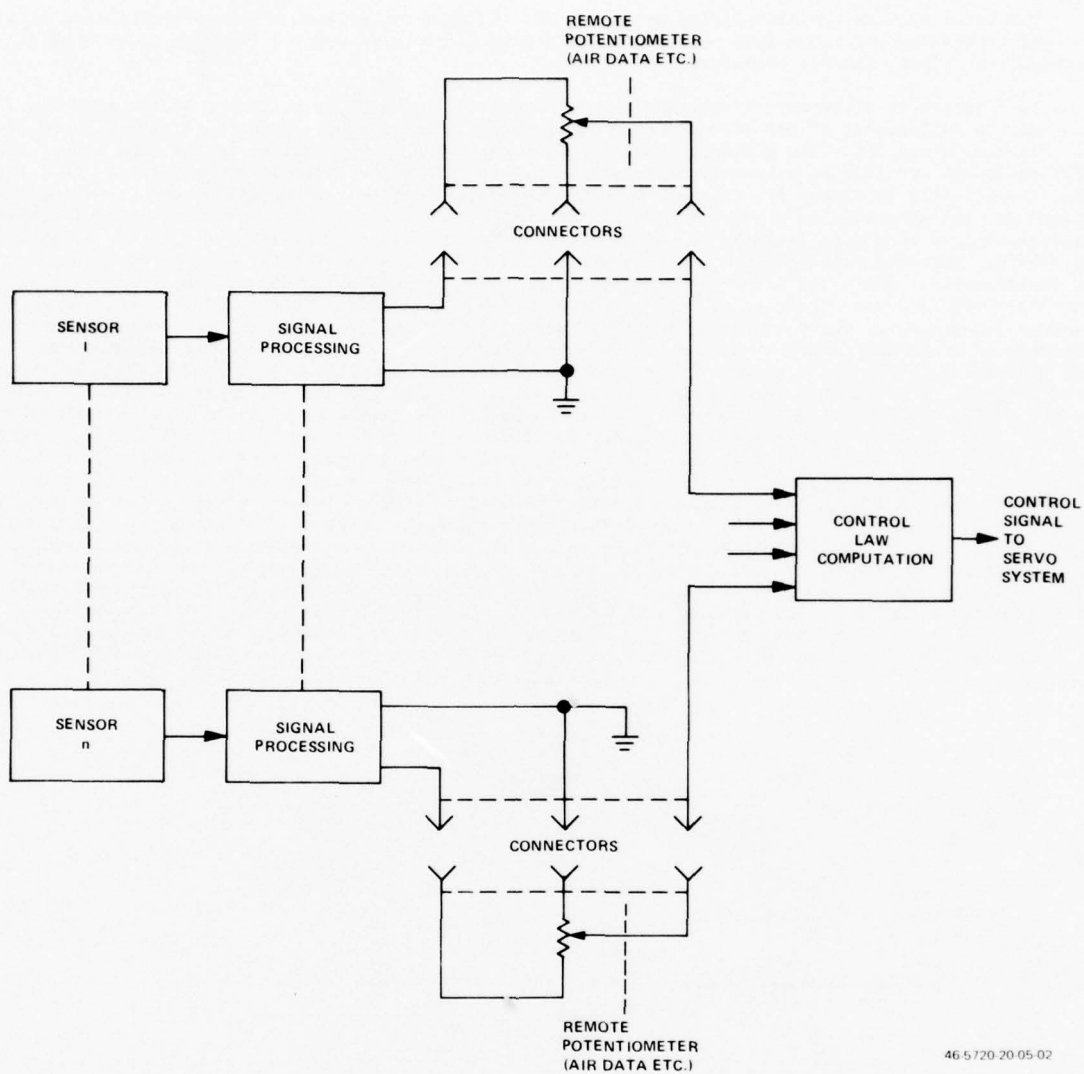
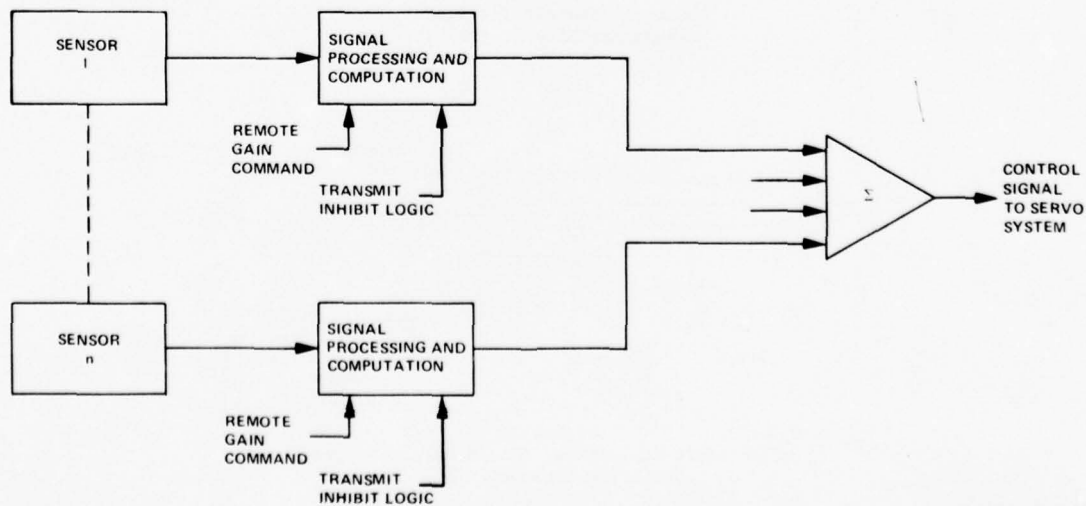


Figure 1  
Evolution of AFCS Control Computer Organization (1958 - 1968)



46 5720 20 05 02

Figure 2a  
System Gain Control Mechanization with Potentiometer  
Multipliers in Signal Path (1958)



46 5720 20 05 03

Figure 2b  
System Gain Control with Internal Electronic  
Multiplication (1963 to 1966)



In the later systems, as exemplified by Figure 2b, the gain control means are self-contained within the signal processing and computation modules which accept remote gain control commands as well as signal transmit or signal inhibit commands.

Another important electronic innovation that affected the internal organization of the AFCS computers was the replacement of the electromechanical instrument servomechanisms with all solid-state devices (Figures 3a and 3b). The electromechanical units that were very prevalent in the 1958 generation of AFCS equipment provided an extremely versatile computational and data handling capability. As illustrated schematically in Figure 3a, they can be used for synchronization, integration, and filtering functions as well as providing a means for mechanically activated limiting and level detection functions. Mechanical devices that were included in electronic computers, however, fell into poor repute in the early 1960's. Improved reliability and maintainability objectives motivated the development of solid-state replacements. While the electromechanical device could perform a variety of functions, only the synchronizer function was difficult to reproduce economically in a solid-state mechanization. Since it required a long-term, drift-free memory, the first versions used analog-to-digital converter registers or counters to store the information, and digital-to-analog converters to complete the synchronizer signal processing loop. Although such conversion devices became commonplace by the mid-1970's as a result of progress in monolithic integrated circuits, in the mid-1960's these converters were large and expensive. They showed little improvement in size and power consumption over miniature electromechanical units, and resulted in a definite cost penalty. They were used only to improve reliability. Were the electromechanical units really so unreliable? The answer depends upon the design application and the quality of the maintenance. This theme recurs as we examine the reliability history of all types of components used in flight control systems. The same type of device commonly gives one to two orders of magnitude difference in field MTBF because of design deficiency and/or poor maintenance. Since reliability of electromechanical computing units are so sensitive to poor maintenance, their replacement with solid-state devices has generally been applauded. By the late 1960's however, the development of extremely high impedance MOS amplifiers in conjunction with special capacitors permitted the achievement of analog memories that actually replaced the digital mechanizations. Thus, a typical solid-state synchronizer-integrator function of 1968, as illustrated in Figure 3b, consisted of an analog-hold circuit, an operational amplifier integrator circuit, a multiplier circuit for line voltage compensation, and switching logic to control the signal inputs to the integrator and the analog-hold memory.

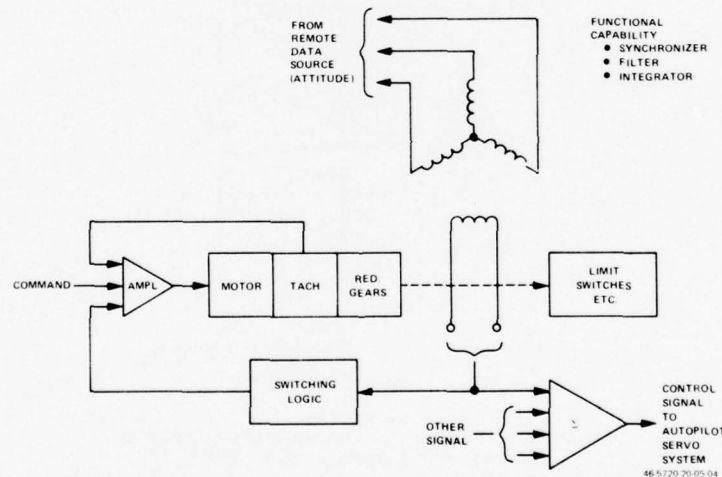


Figure 3a  
Electromechanical Instrument  
Computing Element (1958)

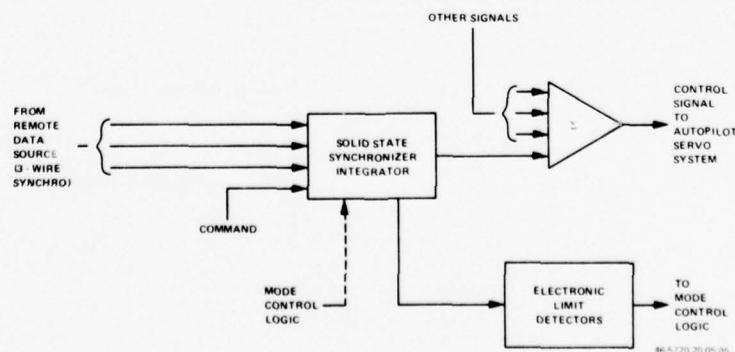


Figure 3b  
Solid State Replacement for Electromechanical  
Computing Element (1965)

The impact of these electronic advances on equipment size is illustrated in Figure 4. A typical 1958 subassembly that required about 950 cubic centimeters of circuit cards had its functions reproduced in 1968 with two microelectronic subassembly embedded modules having a volume of less than 50 cubic centimeters. By 1973, these same functions could be incorporated in two hybrid modules having a

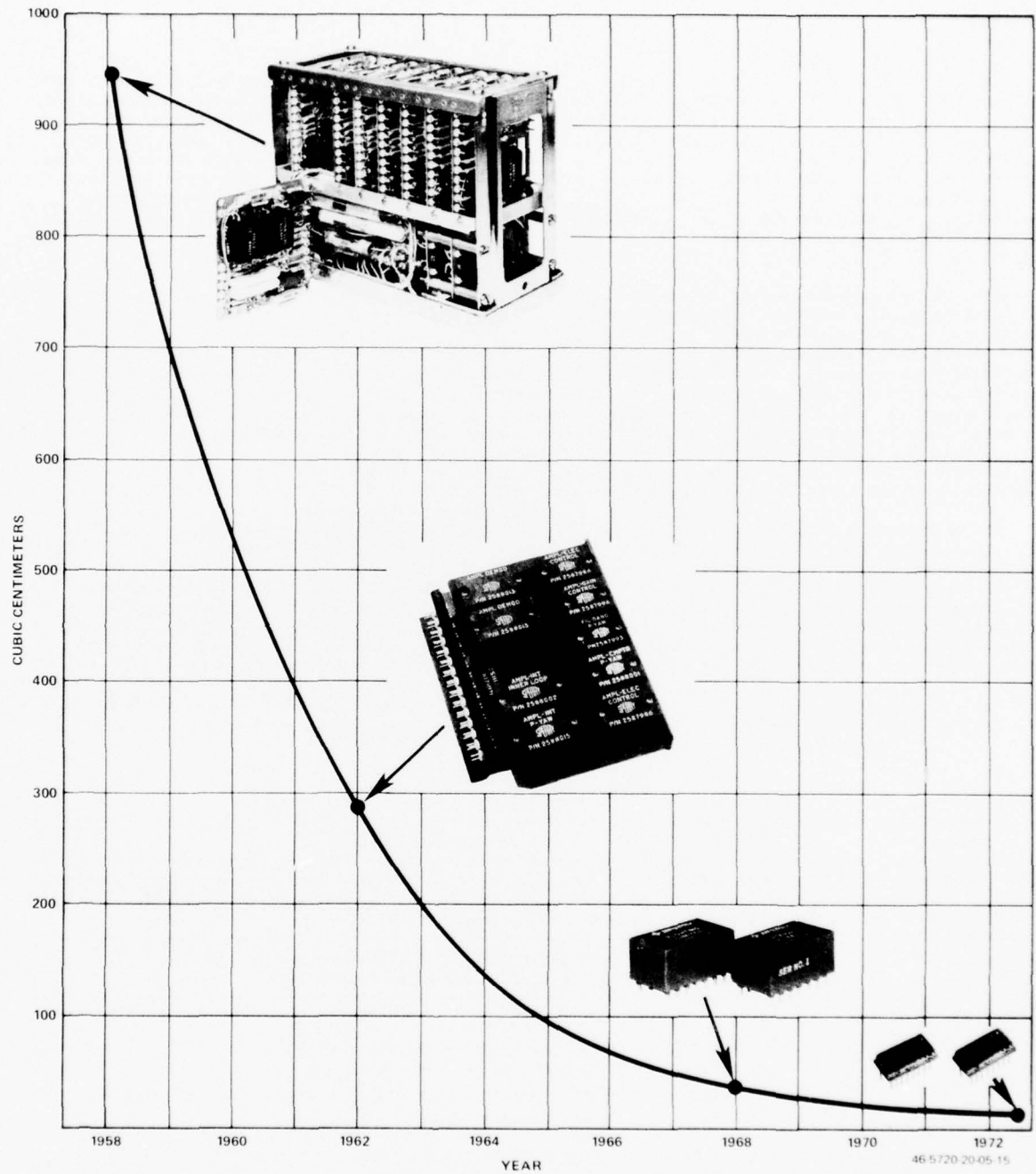


Figure 4  
Miniaturization of AFCS Computer Function (1958 - 1973)

46 5720 20-05 15

volume of approximately one-fifth to one-tenth the volume consumed by the 1968 design. The 1958 starting point on Figure 4 is a large plug-in module from the autopilot used on the U.S. Navy A-6 aircraft. The 1962 module is a control axis for a missile autopilot. The 1968 units are used in the autopilot for the Boeing 747 transport and the 1973 devices are used in the automatic flight controls for the U.S. Air Force B-1.

### 3.0 REVIEW OF SENSOR PROGRESS IN AUTOMATIC FLIGHT CONTROL SYSTEMS

#### A. Overview of Sensors

Progress and trends in flight control sensors are not as easy to track as the electronics. System partitioning has been changing in the past two decades so that sensors that were once part of the flight control systems may now be included within other avionics subsystems. The vertical gyroscope was once the basic autopilot sensor. The attitude-heading reference, which may still use a vertical gyroscope, remains a primary reference for all transport aircraft today but these reference subsystems are no longer included as part of the autopilot. By the mid-1950's, autopilots based on the single degree of freedom, rate measuring gyroscope found their way into all-attitude combat aircraft applications. By the 1960's, the rate gyro became a key sensor for all automatic flight control type systems. It was used in stability augmentors, rate command augmentation systems and as inner loop damping devices for attitude autopilots.

Air data sensors are another example of devices which were once part of the automatic flight controls but eventually grew into independent subsystems. The autopilots of the 1950's contained pitot and static source plumbing connections. They measured static pressure changes from a reference altitude for use in altitude hold cruise modes. They also measured airspeed for use in control law gain adjustment. As transport autopilot guidance modes grew in sophistication to include vertical speed control, airspeed hold and select, Mach hold and select and altitude pre-select, the air data devices contained within the autopilot grew into veritable air data computers. Indeed, in such aircraft as the DC-8 and DC-9, the air data computations performed by the autopilot were expanded into air data computers that supplied display and control information to other subsystems as well as to the autopilot. In most cases however, the aircraft's air data requirements were consolidated into separate air data computers which supplied flight controls and other systems with their required data.

The subject of functional partitioning is important if we wish to enquire into reliability trends in flight control systems because statistical MTBF data for different flight control systems may or may not include the sensors within the flight controls. For example, a review of USAF Manual 66-1 reliability data shows that a vertical gyroscope is a significant failure contributor to some flight control systems (References 2, 3), but airline data on flight control system reliability generally separates vertical gyroscope failure data from that of the automatic flight controls. A similar observation may be made regarding air data devices. Nevertheless, it would be informative to review the technology trends in flight control sensing devices from the standpoint of their impact on reliability.

Table 2 summarizes the various sensing devices used by flight control systems. The following is a brief review of trends in each type.

TABLE 2  
FLIGHT CONTROL SENSORS

Sensor Type	Flight Control Function
Rate Gyroscope	Body axis pitch, roll and yaw rate measurement - used for stability augmentation systems, command augmentation and source of damping for attitude autopilots
Displacement Gyroscope or Platform	Attitude reference (vertical gyro) pitch and roll attitude; heading reference... (Magnetically slaved directional gyroscope)
Linear Accelerometers	<ul style="list-style-type: none"> <li>• Lateral - turn coordination</li> <li>• Normal - pitch command augmentation systems (blended with pitch rate); inertial smoothing of air data derived vertical speed</li> <li>• Longitudinal - inertial smoothing of airspeed measurements for autothrottle controls</li> </ul>
Angular Accelerometers	Integrated to obtain body axis rate measurement - replace rate gyros
Stick Force Sensors	Pilot control input for command augmentation and control wheel steering systems
Air Data Sensors and Computers	<ul style="list-style-type: none"> <li>• Altitude, altitude deviation, airspeed, vertical speed and Mach number for vertical guidance modes</li> <li>• Airspeed and Mach for gain programming</li> </ul>
Miscellaneous Position Sensors	Position measurement of primary control surfaces, stabilizer trim, flaps, wing position, etc using LVDTs, synchros; for control law adjustment, gain control, mode control, etc
Miscellaneous Guidance Sensors	ILS, Tacan, VOR, DME, Radio Altimeter, etc for flight path guidance modes



## B. Attitude Rate Sensing

### 1. Rate Gyros

The single degree of freedom, spring-restrained rate gyroscope used for control applications has not changed much in the past 15 years. The miniature gyro, fully or partially floated in a viscous fluid was introduced in the late 1950's. The 1976 devices are about the same size or only slightly smaller than the 1958 units. The main difference is in the self-test improvements which have been incorporated (precise torque coils and wheel speed monitors). The technology emphasis has probably been directed at cost reduction rather than on reliability improvement although there have been some notable reliability advances. Rate gyros with some penalty in size are available with demonstrable MTBFs of 50,000 hours prior to the onset of an increased failure rate probability resulting from wear out. To operate at the 50,000 MTBF level at all times requires preventive maintenance after 10,000 or 20,000 hours. Such a preventive maintenance requirement, however, is viewed as a serious liability. Indeed, maintenance on any mechanical device, especially a delicate gyroscopic device is a key weakness in the reliability of all flight control systems. In U.S. Air Force experience (References 2, 3), large discrepancies in the attained MTBF of a given part can be attributed to the quality of the maintenance actions. Typical rate gyros in USAF inventory yield closer to 600 hours MTBF than 50,000 hours.

### 2. Derived Rate

The acknowledged problems of maintenance and the historically poor reliability achieved by gyroscopic devices stimulated a search for other techniques of aircraft angular rate measurement. One partial solution to the elimination of the rate gyroscope is a return to the techniques used two decades ago. The autopilots of the early 1950's derived pitch and roll rate electronically from the displacement gyroscope's measurement of the pitch and roll Euler angles with respect to a local vertical coordinate frame. Systems still in use in the B-52 and C-130 aircraft are vestiges of that era. When autopilot system gains and control authorities were increased, the resultant expansion of the closed loop system bandwidth exposed a basic defect of the derived rate technique. Excitation of the vertical gyro synchro transducer was obtained from the aircraft's 400-Hz power supply. Poor regulation of that supply and especially voltage modulation in the 1.0 to 10.0 Hz frequency range would be amplified by the derived rate circuitry. When pitch and roll attitude excursions were large this problem was most severe. It manifested itself as control surface jitter, and with parallel servos, the shaking of the column and wheel would be quite objectionable. Various techniques were employed to minimize this problem such as electromechanical instrument servos of the type illustrated previously in Figure 3a. These instrument servos, in rather elaborate combinations, were used to accomplish signal summation electromechanically so that the 400-Hz suppressed carrier signal that was susceptible to power supply noise was always at a null. Since the noise was proportional to the magnitude of that signal, effective noise suppression could be achieved by working around a null at all times.

By the early 1960's, the need to acquire attitude data from some platform configurations that might interject a servo follow-up unit between the gyroscopic element and the autopilot pitch and roll signal discouraged the derived rate approach and most systems resorted to the single degree of freedom rate gyroscope. By the latter 1960's, however, technology improvements arrived to restore the feasibility of the derived rate approach. Not only did improved constant frequency alternators make up for most of the deficiencies of the older motor driven, poorly regulated inverters, but the advent of microelectronic, operational amplifier computation techniques made line voltage multipliers a practical compensation technique for power supply noise. Flight control systems in the DC-10 and L-1011 aircraft, for example, use derived rates for pitch and roll stabilization. The MTBF of the derived rate circuitry is between 200,000 and 1,000,000 hours, which is considerably better than one could obtain by adding any candidate rate sensor. (It is noted that the attitude reference which is the source of the derived rate has a much lower MTBF than the rate circuit electronics, but loss of an attitude reference shuts down the control channel even if that channel had an independent source of attitude rate.)

### 3. Vibratory Gyros

In the last decade, much work has been done to develop a rate sensor that does not depend upon a high speed wheel. Since the announcement of the vibratory gyro in the 1940's and early 1950's (References 4, 5), designers have worked to develop practical devices. Although not literally in the non-moving part category, vibratory devices featuring the elimination of the high speed wheel offer a potential MTBF of 50,000 to 100,000. They suffer from poor null characteristics but in such applications as yaw dampers where high pass filters in the control law are used to exclude the steady-state yaw rate, the poor null deficiency can be tolerated. Although there have been no widespread applications of vibratory rate sensors, they have been demonstrated (References 6, 7, 8) and might find use in future systems.

### 4. Angular Acceleration Measurement

In addition to derived rate techniques, the main competitor to the vibratory gyros for reliable, no high speed wheel, rate sensing is the angular accelerometer. Practical versions of such instruments are finding their way into airline transport autopilots. Autopilots for the DC-9-50 aircraft now incorporate angular accelerometers. Some 727 aircraft have been similarly equipped. These devices suffer from an inability to accurately define the zero or null attitude rate but in their applications thus far, they are used where the attitude rate can be compensated through a high pass filter (washout). It would be difficult to apply these devices to rate command augmentation systems, where good definition of zero rate point is essential.

Angular acceleration measurement and integration of the resulting signal to produce attitude rate is a technique that has been used for more than two decades. Rather than employ an angular accelerometer, which is an extremely difficult instrument to build in a small package, pairs of linear accelerometers were used. (Linear accelerometers summed differentially will read the angular acceleration of

the line joining them - the larger the separation, the higher the effective angular acceleration signal gradient.) A yaw damper mechanized in this way was used in the U.S. Navy A-3D aircraft in the mid-1950's and all the autopilots in the DC-8 aircraft measure pitch, roll and yaw angular acceleration in this manner. In addition to the DC-8 aircraft, the Convair 990 and 880 aircraft were equipped with similar 3-axis angular acceleration based attitude stabilization systems (References 9, 10, 11). The linear accelerometers used in these applications were of the simplest open loop type consisting only of an E-pickoff and a pair of flexure springs supporting the sensitive mass. It is difficult to find any record of removal of these accelerometers in the last ten years and it is estimated that their MTBFs are in excess of 100,000 hours.

#### C. Attitude Sensors

The trends in attitude sensor size and weight have not followed the electronic trends in regard to miniaturization. In the last two decades, gyroscopic technology advances were in the area of performance improvements required for inertial navigation rather than for flight controls. In the early 1950's, a vertical gyroscope that provided roll and pitch references for an autopilot weighed about 3.2 kg. A modern day, wide-bodied transport aircraft that uses a vertical gyroscope rather than an inertial platform as its source of attitude data has not enjoyed any apparent benefits of miniaturization because its vertical gyroscope unit weighs about 6.8 kg. The unit, however, is packed with electronics for monitoring, signal isolation, erection control and other new features which are demanded by the more sophisticated avionics systems of the 1970's. Reliability of such units is very dependent upon the quality of maintenance. U.S. Air Force data in References 2 and 3 reveal vertical gyro MTBFs ranging from 250 to 900 hours and these are for gyros that are considerably less complex than the type used in modern transport aircraft. Historically, inertial platforms have shown a much poorer MTBF than the vertical gyroscope. The airline experience with vertical gyros indicate an MTBF ranging from 4,000 to 10,000 hours depending upon maintenance quality and method of calculating MTBF (Reference 12). There is some controversy regarding the calculation of MTBF for gyroscopic devices on the basis of flight hours or operating hours. Typical airline experience shows a ratio of operating hours to flight hours of greater than 2:1. (Higher values of MTBF are seen when operating hours rather than flight hours are used in the calculation.) It is noted, however, that most reliability data on modern day automatic flight control systems do not include attitude reference failures as part of the flight controls.

#### D. Accelerometers

Modern units are miniature designs of the feedback type in which integral electronics control the torquer which captures the sensitive mass and processes the signal that measures the acceleration. Reliability histories obtained from field data are often inconclusive because of problems which may have been unrelated to the accelerometer mechanism but caused by the application. Manufacturers of accelerometers are willing to guarantee 100,000 hour MTBFs but quite often the associated electronics detract from the intrinsic reliability of the accelerometer mechanism. Nevertheless, the reliability trend in accelerometer technology for flight control applications is favorable.

#### E. Stick Command Sensors (Force or Position)

These types of sensors which are used for control augmentation or control wheel steering inputs to control computers are mechanized today with strain gauge transducers or LVDT transducers. The transducer devices themselves have MTBFs of well over 100,000 hours. The associated electronics and sometimes installation design deficiencies which aggravate alignment and redundant channel tracking accuracies are the usual detractor from the 100,000 to 500,000 hour MTBF potential.

#### F. Air Data Sensors

When air data sensors are used exclusively for flight control computation, their relative simplicity has resulted in typical MTBFs of 5000 to 10,000 hours for the complete set of air data computations used by a sophisticated autopilot system. However, when the air data computations are centralized so that a central computer provides all the air data needs of the avionics computers, displays and flight controls, then the growth in complexity has led to a decline in MTBF to typical values which are below 1000 hours for commercial aircraft and less than one half that value for some military applications. The trend to digital air data computers which eliminate the complex electromechanical computing and calibration devices that typified the analog air data computers of the 1960's has reversed this trend. Newer central air data computers can be expected to yield MTBFs in the 1500 to 2500 range, depending upon complexity and maintenance factors.

#### G. Miscellaneous Guidance Sensors

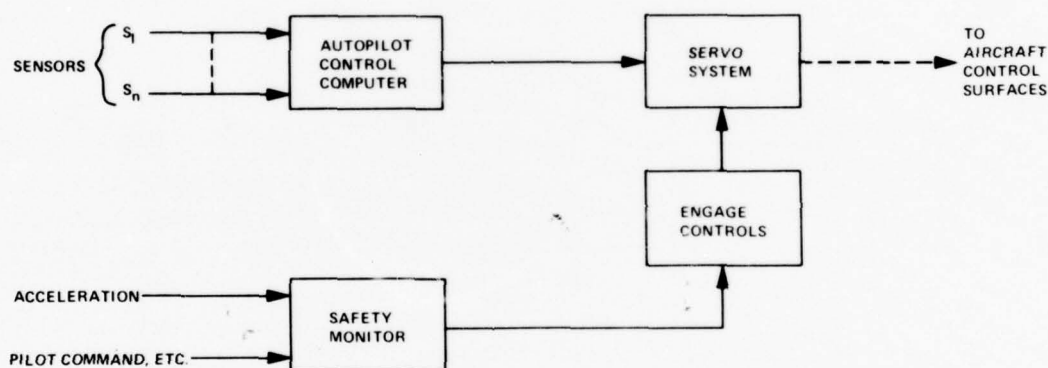
In the compilation of system reliability data, these devices are normally included in categories other than flight controls. Nevertheless, most contemporary automatic flight control systems for transport aircraft include many guidance modes such as the automatic approach and landing functions which depend upon such sensing elements as VHF Nav receivers (VOR, localizer, glideslope), radio altimeters and the various controllers and data entry devices and panels associated with setting up the navigation/guidance problem. This group of devices tends to have operational MTBFs which are of about the same magnitude as the autopilot computers (Reference 13, for example). Consequently, the failure of these radio devices and associated controllers can reduce the probability of equipment availability for an automatic landing to about the same extent as it is reduced by a malfunctioning autopilot computer. In recognition of this fact, there have been recommendations that simplified landing system receivers be developed for exclusive use of the autoland system.

#### 4.0 SAFETY AND REDUNDANCY AND THEIR INSATIABLE APPETITE FOR ELECTRONICS

##### A. Safety Monitors

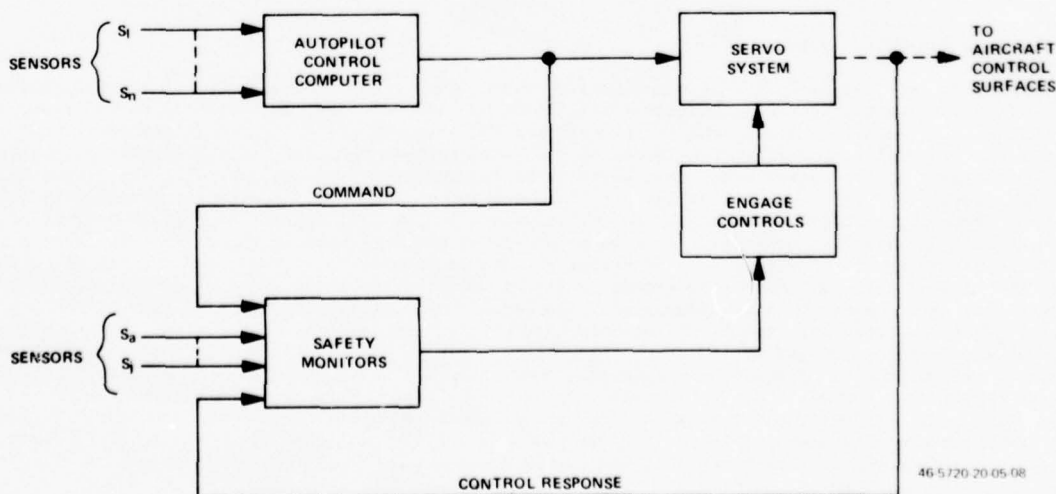
With the progress in miniaturization shown in Figure 4, one is led to inquire how the overall automatic flight control systems have fared in regard to size. There is little basis for comparison with the earlier systems because the scope of functions provided by the automatic flight control systems has been expanding at a continuously increasing rate. This escalation in functional scope appears to be a regenerative process whereby each new function sows the seeds for a future growth requirement. Let us briefly review how this situation came to pass.

Early autopilots were pilot relief devices. They were allowed an absolute minimum of control authority so that they could be overpowered by the pilot with ease and would not cause any serious disturbance if they failed hardover. As their potential for aircraft stabilization and precise flight path control was recognized, it was apparent that performance could be improved with increased static and dynamic authority. This led to a conflict with safety considerations and the protective devices sometimes referred to as Safety Monitors arrived on the scene in the early 1950's. Figure 5a illustrates how the simple mechanization of an acceleration limit detector served as a protective device for an autopilot. The criterion of failure is excessive acceleration, 1.2 incremental g for example. If the autopilot fails hardover, the monitor disengages the system when 1.2g are reached. This system has a natural inclination to grow more complex because it does not do a very good job. If we wait until 1.2g are reached to trip the monitor, we may reach 2.2g by the time disengagement occurs. What if normal control maneuvers reach 1.2g? We do not wish to have nuisance disengagement so we measure the command and modify the g threshold accordingly. What if the 1.2g occur because of turbulence transients? We can compensate for this by measuring and computing whether the autopilot is correcting for the error or aggravating the error. This leads to the more sophisticated Safety Monitor of Figure 5b, where additional sensors and monitor points provide the needed intelligence for failure detection.



46-5720-20-05-07

Figure 5a  
Simple Safety Monitor



46-5720-20-05-08

Figure 5b  
More Sophisticated Safety Monitor



### B. The Dual, Fail-Passive Systems

Soon the requirements for the Safety Monitor become more stringent. Is a transient as high as 1.2g satisfactory, especially if the aircraft is difficult to handle or flying near the ground? Can we reduce the failure transient to .5g, .1g, or .05g? The answer is always yes if we are willing to build a more sophisticated monitor. The ultimate evolution of the Safety Monitor is the Dual Fail-Passive System shown in Figure 6. It determines whether the autopilot's control action is correct by sensing the same information, performs the same computation and commands a response of its own. It does this autonomously of the original control channel. It is, in effect, a complete duplication of the single channel autopilot. To assure a minimum failure transient, the output of the monitor channel can drive a physical servo that actually opposes the failure in the other channel. The criterion of failure detection is a disagreement of the two outputs. The response to a failure detection is total disengagement. What have we created? In the interest of safety, we have introduced redundancy and have thereby reduced reliability by a factor of greater than two-to-one! Here is an enigma that contradicts the instinctive feeling that reliability and safety are consequences of each other.

Now let us examine some of the technological consequences of the monitoring concept illustrated in Figure 6. The main design challenge relates to establishing the failure threshold criteria. Should the channel comparator be set to indicate a failure for 1.0, 5.0, 20 or 50 percent discrepancy between channels? Obviously, the answer depends upon the normal tolerances of a control channel. If we wish to minimize failure transients, the comparator trip levels must be set for small discrepancies. This necessitates tight channel tracking and high accuracy. Indeed, this is the only reason for high accuracy in an automatic flight control system. The control law of a closed loop control process can usually vary  $\pm 30$  percent, statically and dynamically, with little change in performance. Accuracy in the control process is defined by the sensitivity of the sensors and the absence of null offsets in the sensors and control electronics. However, for accurate monitoring, an order of magnitude increase in the precision of the electronics is required. The advances in analog electronic technology during the 1960's permitted the attainment of this type of accuracy improvement.

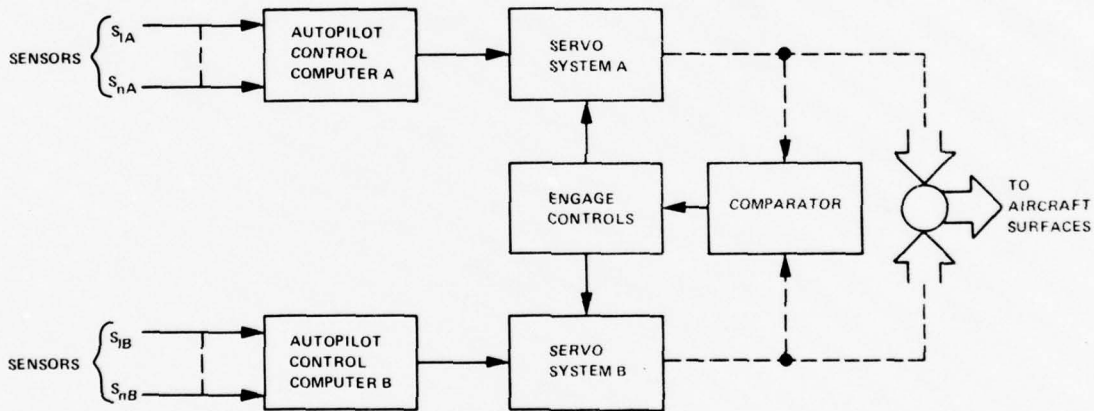
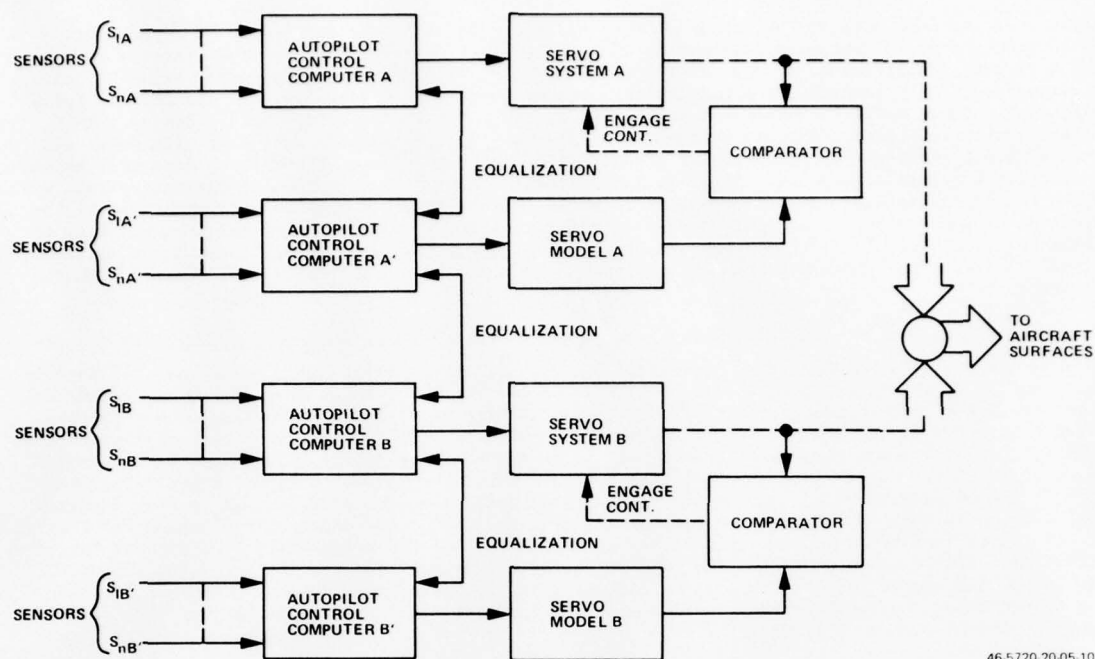


Figure 6  
The Safety Monitor Grows to the Dual  
Fail-Passive System

### C. The Fail-Operative Systems

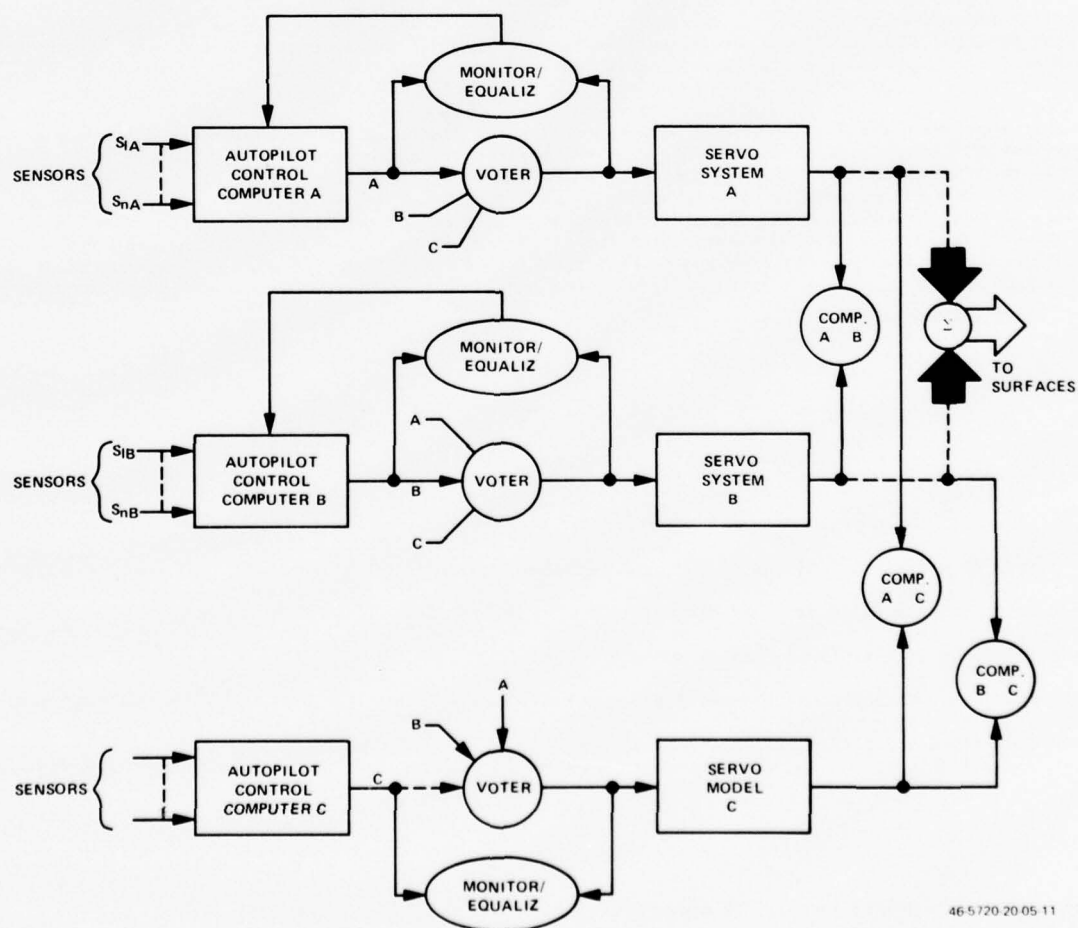
Concurrent with the evolution of a monitoring channel into a completely redundant channel, there were growing demands for increased mission reliability. An aircraft that is committed to automatic control during flight-critical situations cannot tolerate the shutdown of the system in the event of a malfunction. In such automatic control modes as low level terrain following or all weather automatic landing, shutdown of the system even for a short instant is not permissible. The need for proven solutions to this type of problem has been the principal deterrent to the introduction of fly-by-wire controls and their acknowledged advantages. Hence, the need for the fail-operative or fault correcting system arose. Its simplest mechanization from a conceptual point of view is the direct growth from the dual fail-passive system of Figure 6. By taking two dual fail-passive systems, we have the brute force growth to the dual-dual fail-operative system of Figure 7a. Either channel A or B can perform the control task and each channel monitors itself with its redundant monitor. If channel A fails, it shuts down, gracefully recentering itself, and permits channel B to continue the control task. There are a number of complications suggested in Figure 7a. First, one might not wish to build quad redundant hydro-mechanical servos because of weight and hydraulic power redundancy considerations (although such systems are operational today). Thus, electronic models of the servo are often used for monitoring purposes. Also, two completely autonomous, active control channels cannot control the same aircraft without certain types of run-away problems. The two channels must be equalized or slaved to each other. This is a dangerous requirement because it can permit cross-channel failure propagation.

In principle, it should not require four channels to produce a fail-operative system. A common monitor channel can be used to identify the malfunctioned channel by a two out of three voting procedure. Figure 7b shows this triple redundant fail-operative configuration, but it also shows another interesting



46 5720 20-05-10

Figure 7a  
The Dual Fail-Passive System Grows to the Dual-Dual  
Fail-Operative System



46-5720 20-05-11

Figure 7b  
The Triple Redundant Fail-Operative System

innovation that has been used in the design of such systems. The input to each servo or servo model channel is always a valid control signal and is always identical in all channels. This feature is provided by a circuit identified as a voter in Figure 7b. It is often referred to as a Mid Value Signal Selector circuit. It only transmits the middle value of the three channel signals. If one channel fails hardover, that failure would be completely suppressed by the mid-value circuit. Such a device permits some relaxation in tolerance tightness because it acts as a signal collection node that eliminates all tolerance errors up-stream. The down-stream monitoring of the servo system, therefore, need only be concerned with servo system tolerances. Also, failure detection circuitry up-stream of the voting node may have wider thresholds because the failure suppression is inherently and instantaneously provided by the voting circuit. But a note of caution: Any node that represents a convergence of multichannel data is vulnerable to cross channel failure propagation which can allow a single failure to wipe out a multichannel system. These nodes, therefore, must be properly buffered, monitored, tested, and treated with the utmost care.

The problem of safety at the multi-channel convergence nodes becomes even more critical in the so-called (fail-operative)<sup>2</sup> or double fault correction system. After any first failure, it must continue to operate normally and reject any second failure without transient disturbances or performance degradation. In the latter 1960's, development programs for such systems were underway. The motivation was fly-by-wire. Two programs typified how the analog technology of the late 1960's coped with the (fail-operative)<sup>2</sup> requirement for military and commercial operations: The USAF 680J Survivable Flight Control System in the F-4 aircraft (Reference 14) and the U.S. Supersonic Transport which was being built by Boeing (Reference 15). The complexity of the electronic mechanizations in these two programs was largely dictated by safety considerations. Where safety constraints were more severe, the electronic solution resulted in greater complexity (and hence reduced reliability in the sense that the probability of equipment failures had increased). Reference 16 shows how the more stringent requirement for protection of the signal voting node in the SST system resulted in a voter circuit that used four times as many components as was required to mechanize the voting circuit used in the F-4 Survivable Flight Control System. This complexity growth is in addition to a factor of 2 to 3 increase in the complexity of a 4-channel voter circuit over a 3-channel voter.

#### D. Complexity Growth in Analog Fail-Operative Systems

The subject of the voter circuitry is important because it became a key issue in the design philosophies which were incorporated into the fail-operative type systems being developed in the latter 1960's. The voting or signal selection circuit is a generalization of the majority logic algorithm illustrated in Figure 8a for three discrete inputs A, B, and C. The digital majority logic mechanism is usable directly as a signal selector when the signal is transmitted as pulse width modulation (F-111 Automatic Flight Control System, for example). In the more typical case of contemporary analog control systems, the signal is dc or suppressed carrier ac. In such cases, a circuit which selects the most positive signal is substituted for the "or" gate and a circuit that selects the most negative signal is substituted for the "and" gate. When there are four channels, several other arrangements of the same elements allow the selection of one of the two middle values. One such arrangement is shown in Figure 8b. That version, mechanized with nine amplifiers plus additional circuitry is used as a most negative mid-value signal selector in the L-1011 flight control system. Circuits that perform this type of signal selection appear to be ideal methods of combining the signals from redundant sensing or computation channels. They should have allowed the mechanization of system architectures that employ sectionalized redundancy to enhance mission success. For example, consider a quad redundant system consisting of 4 sensors, each having a failure probability of  $Q_1$ , and 4 computers, each having a failure probability of  $Q_2$ . Using the binomial distribution formula (Reference 17):

$$P_F(r/n) = \sum_{x=r+1}^n \frac{n!}{x! (n-x)!} Q^x P^{(n-x)} = \text{Probability of Failure}$$

where

$r$  = number of allowable failures (3 for quad redundant system)

$n$  = number of channels (4 for Quad System)

$P$  = probability of individual channel success

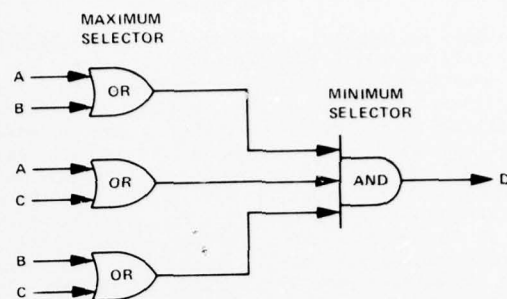
$Q$  = probability of individual channel failure =  $1 - P$

It is seen that cross strapping the individual sensors to each computer yields a  $P_F$  of:

$$(P_F)_{\text{CROSS STRAPPED}} = Q_1^4 + Q_2^4$$

while connecting each sensor to only its associated computer channel yields a  $P_F$  of:

$$(P_F)_{\text{ISOLATED}} = (Q_1 + Q_2)^4$$



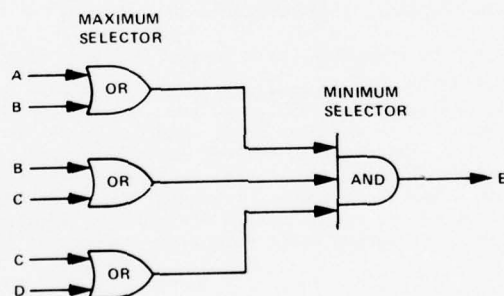
$D = (A+B) \cdot (A+C) \cdot (B+C)$   
FOR DISCRETE LOGIC

$$D = \text{MIN} \left\{ \left[ \text{MAX} (A,B) \right], \left[ \text{MAX} (A,C) \right], \left[ \text{MAX} (B,C) \right] \right\}$$

FOR ANALOG MID VALUE SELECTION

46-5720 20-05 12

Figure 8a  
Generalization of Majority Logic to Analog  
Mid Value Selection



$$E = \text{MIN} \left\{ \left[ \text{MAX} (A,B) \right] \left[ \text{MAX} (B,C) \right] \left[ \text{MAX} (C,D) \right] \right\}$$

46-5720 20-05 13

Figure 8b  
Typical 4-Channel Mid Value Voter



If the sensor and computer have equal failure rates ( $Q_1 = Q_2$ ) then the ratio of isolated channels to cross strapped channel failure probability is 8:1. Even if the sensor has only one-tenth the failure of the computer ( $Q_1 = .1Q_2$ ), the ratio of isolated to cross strapped failure probability is 1.46:1.

Despite this apparent advantage, sectionalized redundancy through cross strapping of sensors was generally not implemented in the fail-operative redundant systems of the late 1960's and early 1970's. Mechanization complexities (not visible to the reliability theoreticians who calculated the advantages using the above type of analyses) discouraged this practice. The mechanization device was the signal selector or voter. In addition to the requirement that voting circuits provide proper electrical isolation to prevent a failure at the voter from propagating back into the input signal sources, faults of input signals must be detected and reported and in many cases (the quad voter, for example), the voter circuit must be reconfigured to cope with the surviving inputs. Perhaps the most serious contributor to complexity growth was the need for special self-test circuitry. Because of the voter circuit's inherent failure suppressive properties, it suffers from latent failure vulnerability. Not only does it suppress failed input signals, it suppresses many of its own failures. When such latent failures exist, the circuit acts very normally until it is actually needed to perform its failure suppressive role. One of the input channels goes hardover and the mid-value circuit, because of its undetected latent failure, transmits the failed signal rather than the good ones. Moreover, the failure detection logic is confused and it shuts down the wrong channel. Obviously, such a situation is frowned upon even though the probability of such a combination of events is very remote. How does one correct for this problem? We must add a means of exercising the internal structure of the mid-value logic circuit so that any latent failures can be detected and reported. This added complexity often involved more circuitry than we had in the device being tested; and then, the test circuitry failure peculiarities must be considered in great detail.

A complexity divergence became apparent to designers of analog redundant systems in the latter 1960's. Figure 9 was developed by analyzing the number of circuit components used for redundancy management functions in typical redundant system designs of the 1968 era. The figure shows that the circuitry needed to perform the control and logic functions associated with aircraft stabilization and control becomes an insignificant part of the total electronics. The redundancy management electronics which provided the circuitry for accuracy enhancement, fault isolation, fault reporting, and built-in test became the dominant part of the systems. The more sophisticated of these systems began to incorporate small digital computers to perform detailed pre-flight checkout of the redundancy management electronics. This checkout was essential to maintaining mission success probability because the redundancy management system was effectively a fault correcting mechanism which tended to mask failures. The latent failure, such as that discussed above for the voter failure, had to be uncovered by a form of preventive maintenance. The vulnerability of a quad redundant system to such types of failure is illustrated in Figure 10. This figure is reproduced from Reference 16 which analyzed the latent failure effects of a quad redundant system with each channel having a 1600 hour MTBF. The figure shows probability of a one hour mission success when the in-flight failure monitoring capability is only 90 percent and various degrees of pre-flight Built-In Test (BIT) thoroughness are employed. It is seen that without the thorough BIT performed as a pre-flight, mission reliability deteriorates excessively.

The final strains on analog system complexity were reached in this 1967-1970 era. In the experimental quad-redundant (fail-operative)<sup>2</sup> systems, separate digital computers were being employed to perform this BIT. In the USAF Survivable Flight Control System in the F-4 (Reference 14, 18), a separate BIT digital computer and associated Maintenance and Test Panel were used for this function. In the NASA Digital Fly-By-Wire system in the F-8, the triplex, back-up electronics unit incorporated a special purpose digital computer within its control and display panel to perform the pre-flight BIT that tested the triplex electronics as well as the primary and back-up redundant hydraulic actuators. In the U.S. SST program, the quad redundant fly-by-wire and stabilization electronics (Reference 15) were going to be tested by the general purpose digital computers which formed the nucleus of that aircraft's digital autopilot.

#### E. Digital Systems

One message became very clear to the redundant system designers of the latter 1960's. As we tried to guarantee a redundant system's safety by correcting for circuit or logical defects in the monitoring and redundancy management, more hardware had to be added. Adding hardware, in turn, complicated the BIT and its interface with the remainder of the system. This complexity spiral could be eliminated if we built the systems around a general purpose digital computer. That computer would not only perform the control law and logic functions needed to stabilize and control the aircraft but it could perform the monitoring, redundancy management and BIT without adding any more hardware than what is needed for the basic control function. Once the interfaces with the sensors and actuators are established, system test, fault isolation and reporting can be accomplished in the software. Such systems have been demonstrated in commercial transport applications (Reference 19) and military aircraft applications (Reference 20). These demonstrations have verified the breakthrough in built-in system testing, fault isolation and fault reporting which can be obtained in a properly designed digital system. The potential advantages that these new systems can provide in the area of maintenance management cannot be underestimated. One of the problems that plagues the maintenance of contemporary analog redundant flight control systems is a monitor activated disconnect in flight which cannot be traced to any faulty component when the aircraft returns. The digital system can record, into a non-volatile memory, the specific cause of an in-flight disengagement so that ground maintenance actions can concentrate on the isolated faulty component. Since much of the historically poor reliability record in avionics is attributed to improper LRU removal and consequent maintenance actions on the wrong box, the properly designed digital system offers promise of alleviating this problem.

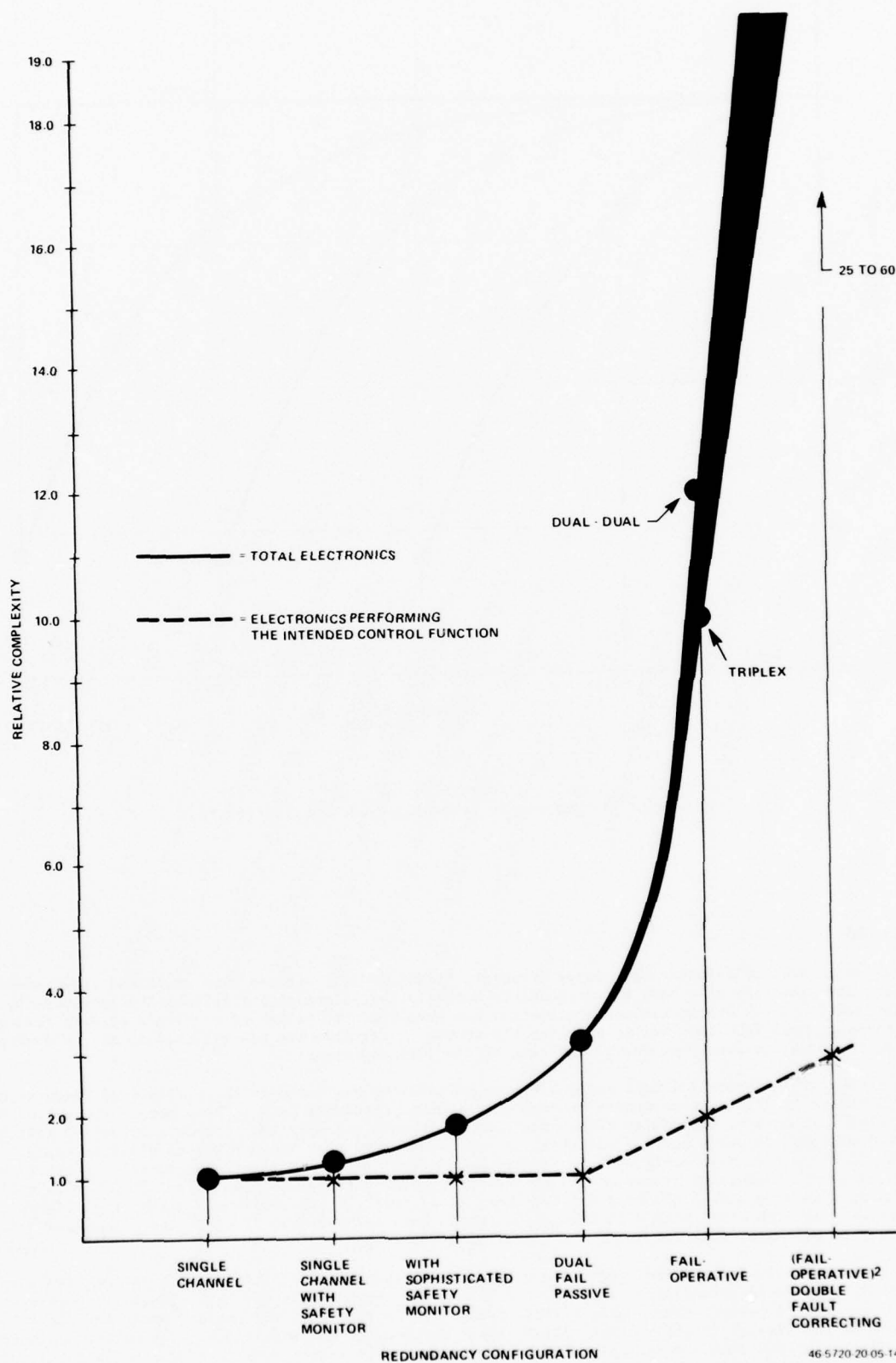


Figure 9  
Effect of Redundancy Configuration  
on Equipment Complexity

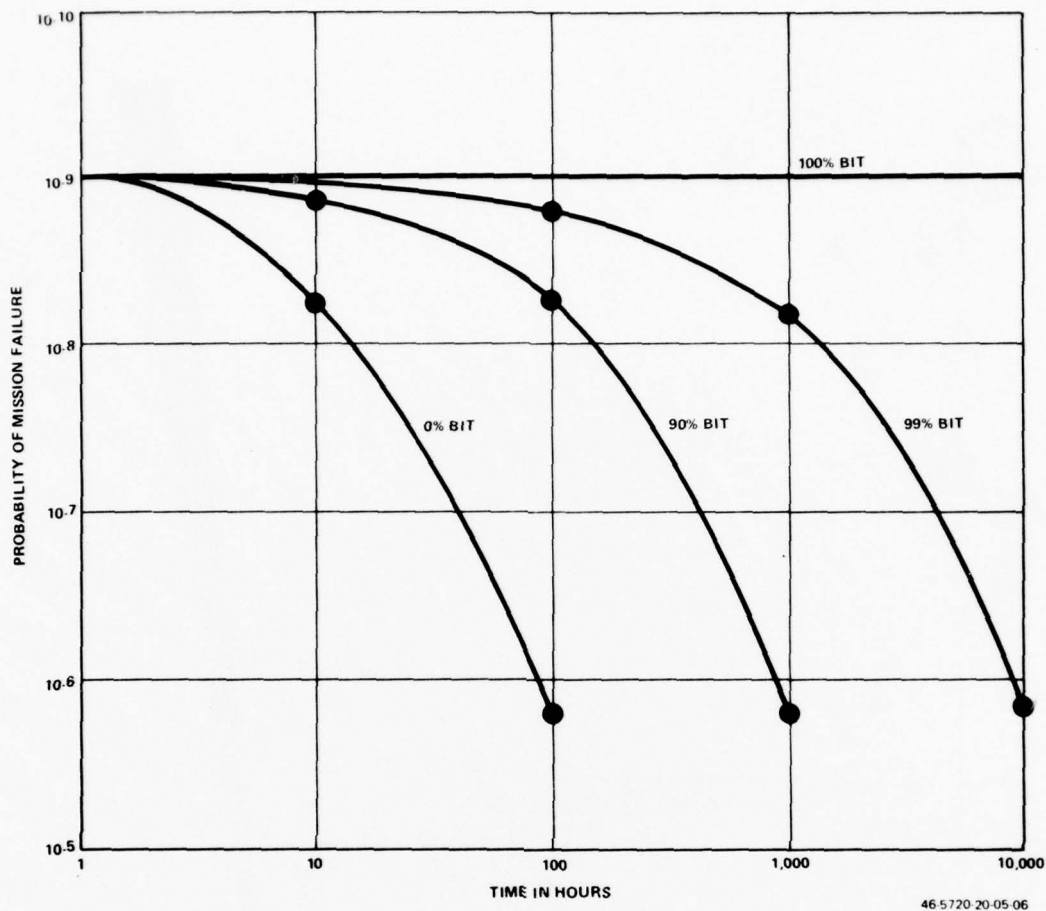


Figure 10  
Effect of BIT Testing on Probability of Mission Failure

#### CONCLUSIONS

In two decades of dramatic technology advances, flight control systems have exploited these advances by adding new functions at a rate which tended to outstrip the potential for reliability improvement. In the past twenty years, electronic miniaturization has permitted two orders of magnitude of size reduction for a given control function, but in practice the growth in requirements has resulted in flight control electronics which are about one-fourth the size of the 1956 equipment.

A review of how overall flight control system reliability has fared in the presence of these technology advances is difficult to assess because maintenance procedures tend to have more influence on reported MTBF than much of the design technology used to improve reliability. Complete autopilot systems that were designed 20 years ago and are still in use today often show field MTBFs of under 50 hours. An MTBF of 100 hours for these early systems (including vertical gyros and electromechanical actuators) was considered good. Commercial transport flight control systems designed in the latter 1960's, when well maintained, typically give MTBFs of about 400 hours (less vertical references and actuators). These same systems, if mechanized with 1958 technology, would consume more than 10 times the volume and result in an MTBF of much less than one-tenth the values achieved in the 1970's.

The strongest impetus to the increasing complexity of systems originating in the latter 1960's was redundancy, (motivated primarily by safety), and its unique requirements for built-in test. Such systems pressed the best of contemporary analog circuit technology to the limit and set the stage for the development of the digital flight control systems which appeared by the mid-1970's. The main contribution of the digital system from the standpoint of reliability will be its ability to handle monitoring, redundancy management and built-in test without adding more hardware. This offers significant promise in the area of maintenance management since maintenance procedures have been shown to be the most significant factor in field MTBF for both commercial and military systems.

## REFERENCES

1. "Proceedings of the Self-Adaptive Flight Control System Symposium (P. C. Gregory, Editor)," WADC TR 59-49.
2. Bielka, R. P., Kavalok, A. P., Johnson, W. L. and Reel, R. E., (Boeing Company), "Aircraft Flight Control Systems Field Safety Experience," AFFDL-TR-72-33, May 1972.
3. Air Force Manual 66-1
4. U. S. Patent 2,309,853, Re. 22,409, "Rate and Attitude Indicating Instrument," J. Lyman, December 21, 1943.
5. U. S. Patent 2,513,340, and 2,627,400, "Angular Velocity Responsive Apparatus," J. Lyman, July 1950 and February 1953.
6. Klass, Philip J, "Vibrating-Beam Rate Sensor Tested," Aviation Week and Space Technology, January 12, 1970.
7. U. S. Patent 3,504,554, "Vibrating Wire Angular Rate Sensor," April 7, 1970.
8. Gates, W. D. "Vibrating Angular Rate Sensor May Threaten The Gyroscope," Electronics, June 10, 1968.
9. U. S. Patent 2,808,999, "Automatic Flight Control Apparatus," P. J. Chenery, October 8, 1957.
10. U. S. Patent 3,007,658, "Aircraft Automatic Pilots," H. Miller et al, November 7, 1961.
11. Miller, H. and Wagner, R. H., "Flight Control System for Jet Transports," IRE Transactions on Aeronautical and Navigational Electronics, September 1957.
12. Sperry Flight Systems Airline Equipment Removal Report, May 12, 1976.
13. United Airlines Premature Removal Reports (SFOMI-Maintenance Information Systems), Any 1976 Monthly Report for 727, 737, 747, DC-10 fleet.
14. Hooker, D. S. et al, McDonnell Douglas Corporation Survivable Flight Control System, Final Report, 1973, AFFDL-TR-73-105.
15. Tomlinson, L. R., "Control System Design Considerations for a Longitudinally Unstable Supersonic Aircraft," Journal of Aircraft, Vol. 10, No. 10, October 1973.
16. Flannigan, J. and Emfinger, J., "Production Design Requirements for Fly-By-Wire Systems," AGARD Conference Proceedings No. 157 on Impact of Active Control Technology on Airplane Design, October 1974.
17. "Handbook, Reliability Engineering," NAVWEPS 00-65-502, Appendix 2, Reliability Formulae
18. Krachmalnick, F. M., et al, "Survivable Flight Control System - Active Control Development, Flight Test, and Application," AGARD Conference Proceedings No. 157 on Impact of Active Control Technology Avionics System Conference, April 1975.
19. Osder, S., Mossman, D., and Devlin, B., "Flight Test of a Digital Guidance and Control System in a DC-10 Aircraft," AIAA Paper 75-567, AIAA Digital Avionics System Conference, April 1975.
20. Honeywell Inc. "Flight Test Evaluation of a Digital Multimode Flight Control System for the A-7D Aircraft," AFFDL-TR-75-97, November 1975.



SAFETY CRITERIA FOR FAIL-OPERATIONAL AUTOLAND SYSTEMS  
AND THEIR APPLICATION

D.V. Warren

Airworthiness Division  
Civil Aviation Authority  
Brabazon House  
Redhill  
Surrey RH1 1SQ  
U.K.

SUMMARY

The airworthiness requirements for the certification of automatic landing systems in civil aircraft include an explicit statement of the safety level to be achieved. For compliance with these requirements a safety assessment of the system must be made, and accepted by the airworthiness authority. It must contain a logical analysis which identifies all critical failure conditions of the system and shows that the probability of each is appropriate to the degree of hazard associated with it. It should also examine the factors which influence the performance of the system and show by means of analysis, simulation and flight testing that the safety level will be acceptable. The analysis will establish the maintenance checks necessary together with their frequency, and any other limitations on the use of the system.

1. INTRODUCTION

In the United Kingdom the Civil Aviation Authority is responsible for ensuring that all aeroplanes registered in the country are airworthy by virtue of both their design and their maintenance. This responsibility is discharged by ensuring compliance with a published code of airworthiness requirements. Before an aeroplane receives a Certificate of Airworthiness, without which it may not operate, there is an investigation of the design against the yardstick of the requirements. In practical terms the CAA prepares the requirements and provides interpretative material while the manufacturer is obliged to carry out whatever testing and analysis is necessary to demonstrate compliance to the satisfaction of the CAA. The aim of this paper is to describe those particular requirements which refer to automatic landing systems, their background and application. To date the following aeroplane types have been approved by CAA for automatic landing against these requirements: BAC 1-11 and SVC-10, Hawker Siddeley Trident, Boeing 747, Lockheed L-1011, Concorde.

2. ACCIDENT RATES AND ACCEPTABLE RISK

In regulating airworthiness it is important to realise firstly that "complete safety" can never be achieved, and secondly that safety is a commodity with a price. Thus the level of safety will be very closely related to the money spent on it, and in turn that money must come out of the cost of the ticket to the passenger. In seeking to determine whether aviation safety levels are acceptable or not, it is fruitless to make comparison with other modes of transport. Aviation can be shown to be "better" or "worse" than motor cars or trains depending on the chosen index (i.e. number of deaths per passenger-mile, or passenger journey etc.). The only useful indication of acceptability is the fact that the public flocks in its millions each year to fly on the world's airlines, in spite of the publicity always accorded to aeroplane accidents. This suggests that aviation safety is at a level which is generally accepted as reasonable in relation to the cost of the ticket.

However, the average layman is not able to assess the risk or the factors that influence it, so that it is a responsibility of the authorities to ensure that safety is maintained overall and that it is reasonably uniform. It is also generally agreed that there should be a steadily increasing level of safety, and this for three main reasons:

- (a) to preserve or improve the position of aviation in relation to other means of transport
- (b) so that there is no increase in the number of accidents occurring in any one year, even though there is a continual growth in the volume of aviation
- (c) so that the safety of a new aeroplane type matches the safety of that which it is replacing (typically the safety of a type improves as it matures and has its major problems resolved).

This is reflected in the fact that for world-wide scheduled air transportation (excluding China and the U.S.S.R.) the number of fatal accidents per million hours flown has decreased progressively from a value of about 4.0 in the early 1950's to 2.3 for 1974. The latter figure corresponds to a value of 2.9 fatal accidents per million flights. It is worth noting that these two indices are the most commonly used in aviation since they have the merit that they relate easily to operational records and equipment failure rates, and are also a measure of the risk to the individual passenger that he will be in a fatal accident.



### 3. AIRWORTHINESS REQUIREMENTS

Most airworthiness requirements are rather dogmatic statements specifying a minimum or maximum value for a particular parameter or characteristic of the aeroplane. Although they appear to be entirely arbitrary with no mention of probabilities or statistical distributions, many have as their basis a rational calculation to give a particular level of safety. For example, the aeroplane is required to survive encounters with gusts of certain magnitudes, which are specified from consideration of their probability. Again, the climb capability of an aeroplane with an engine failure in various phases of flight is based on the probability of such a failure, and the statistical variability of climb gradient for other reasons. However, requirements can only be derived in this way when there is sufficient background and experience to establish the statistics with confidence. When, in the early 1960's, airframe and equipment manufacturers asked us to state the airworthiness requirements which we would apply to an automatic landing system, and particularly to its use in near "blind" conditions, we had little relevant experience from which we could derive requirements. Furthermore there was a danger that if we attempted to write detailed requirements of the conventional kind we might inadvertently place unnecessary constraints on the design of systems, so inhibiting inventive design. In consultation with industry we decided that we should state the requirements in terms of safety levels to be demonstrated by a suitable safety assessment, backed by the necessary analysis and testing. The choice of safety levels, and the form of the safety assessment will be discussed in detail in the paragraphs that follow. Essentially the consequences of system failures must be considered, and also the effect of variations in system performance when there is no failure present.

Although the declared objective of the assessment is to show that the specified safety level is achieved, in practice it is the critical and logical scrutiny of the system which is of most value, and not the precision of the numerical conclusions.

Safety assessment techniques are being more and more widely applied in the certification of aeroplanes, and indeed in some cases to problems which are not as amenable to numerical analysis as is an automatic landing system.

### 4. SAFETY LEVELS REQUIRED FOR AUTOMATIC LANDING SYSTEMS

The main purpose of introducing automatic landing systems was to permit operations in much lower visibilities than had been possible hitherto, with the ultimate objective of landing virtually "blind". The principle was established that the average risk in these new operations should be the same as the overall safety level for all landings in all weathers. In fact if this were achieved, the net result would be a modest improvement in safety because some of the more marginal of the existing operations would be made safer.

In addition it was considered necessary to limit the risk which could be taken on a specific flight so that an unduly hazardous landing would not be undertaken even though average risk considerations would permit it. The maximum risk associated with using the system was limited to approximately the total average risk for a complete flight. This was arbitrary but recognised that a decision to divert is likely to incur the risk arising from the alternate back to the original destination.

In short, the airworthiness requirements are stated in two forms - average risk, and the specific risk at the last point in the intended flight from which a safer diversion can be made. The following is an extract from Reference 1 :-

#### "Average Risk"

The system shall be such that the total fatal landing accident rate (i.e. average risk) due to the use of the system at any time and in the new visibility conditions permitted below current minima (approx. 200ft. and  $\frac{1}{2}$  mile) shall not be greater than the present total fatal landing accident rate for all transport aircraft. This figure is believed to be of the order of one fatal accident per million landings. Since piloting is only one of several possible causes of fatal landing accidents, the system should not contribute a rate greater than  $1.0 \times 10^{-7}$  fatal accident per landing.

#### Specific Risk - Risk on a Particular Flight

##### Take-off

No flight shall be started with the intention of making a landing using the system if conditions are expected to exist such that the risk of a fatal accident due to the use of the system is greater than  $3 \times 10^{-6}$ .

In the foreseeable future, the probability of en-route failure in the system, and the probability of deterioration in weather to a state where the system cannot be used, is likely to be such that it will not be possible to meet this take-off criterion without providing for an en-route option to divert to an alternative destination. In this case, compliance with the  $3 \times 10^{-6}$  take-off risk will consist in ensuring that a suitable alternate is available and that adequate provision is made for the pilot to receive and use such information as he may require to make the decision whether to divert or not. In specifying the alternate, account should be taken of the probable weather conditions there, and the probable state of system serviceability. Adequate fuel reserves should be carried.

### Risk after Take-off

Where the use of the system depends on the provision of an alternate destination the decision to divert should be taken, if during the flight circumstances arise such that the risk of a fatal accident due to the use of the system exceeds  $3 \times 10^{-6}$  and provided that the diversion is likely to be a safer course of action.

The crew must not be expected to calculate the risk but must be given guidance in the form of limitations on such factors as aeroplane or equipment unserviceabilities (including the elements of the system), weather at the destination, and the state of the runway. When limitations are being derived, each one shall be set such that, with that parameter on its limit, the average flight landing accident risk would be less than  $3 \times 10^{-6}$ . It would be permissible to continue to use the system even when several parameters were known to be near their limits, provided they have an unrelated effect. So far as is practicable, however, guidance should be given to the crew so that they would avoid using the system if several parameters were near their limits, and all affected the system in a similar way, e.g. if all tended to increase the variability in touchdown position."

## 5. AUTOMATIC LANDING SYSTEMS

An automatic landing system contains a number of sub-systems and equipments distributed throughout the aeroplane and many of these are used for purposes other than automatic landing. The system design varies enormously from one aeroplane type to another, but essentially it consists of:-

- (a) sensors, normally at least the following:-
  - ILS glide slope and localiser receiver,\*
  - radio altimeter,
  - pitch, roll attitude and rate,
  - air data computer (airspeed and vertical speed),
  - lateral and longitudinal accelerometers, compass.
- (b) computation and control actuators to enable the aeroplane to "couple" to the ILS localiser and glide slope during the approach, to control the airspeed, to remove the drift angle due to cross-wind before touchdown, to reduce the descent rate before touchdown ("flare") and to control the landing run to the runway centreline.
- (c) switches, instruments, warning lights, indicators to enable the pilot to control and monitor the operation of the system.

Equipment may be duplicated where there is a need to detect a failure and shut the system down without causing an unwanted manoeuvre to the aeroplane (fail-passive). Further redundancy is incorporated when there is a need to ensure continued operation after a failure (fail-operational).

## 6. SAFETY ASSESSMENT

Because the requirement is simply a statement of the risk level to be demonstrated, there is an automatic obligation on the manufacturer to produce a safety assessment. Experience has shown that an analysis which simply sets out to examine the effect of all component failures, tolerances, environmental conditions with the infinity of possible combinations will rapidly become unwieldy and ultimately unworkable. It is much more practicable to start at the output end of the system and to consider what hazardous effects the system can generate. It is then possible to allocate an appropriate proportion of the permitted  $10^{-7}$  risk to each effect and to work back through the system finding all the ways by which such an effect can come about. This list of hazardous effects, and the permitted probability for each, may be taken as the "airworthiness objectives" for the system.

\* The ILS ground station is approved separately as operating to specified failure and performance characteristics. It comprises:-

- (i) an ILS localiser transmitter providing azimuth guidance to and along the runway,
- (ii) an ILS glide slope transmitter providing a  $3^\circ$  descent path to the touchdown zone of the runway.

### 6.1. Airworthiness Objectives

Airworthiness objectives may be derived by considering the "freedoms" the system enjoys. In simple terms an automatic landing system which is functioning incorrectly may cause the aeroplane to have:

- (i) Incorrect position, e.g. such that it strikes an obstacle on the approach, or lands off the runway
- or
- (ii) Incorrect rate of change of position, e.g. such that the rate of descent at landing, or the lateral cross track velocity, causes structural damage
- or
- (iii) Incorrect attitude at landing, e.g. high bank angle which causes the wing tip to touch or a high pitch angle causing the tail to scrape.

Not all the hazardous effects can by any means be regarded as certain fatal accidents or catastrophes. Even when there is a high probability of fatality, there may be a sufficient chance of a safe outcome, that credit can be taken for it in the analysis. This would allow the probability of the occurrence to be greater than if it were regarded as being certain to cause a catastrophe. CAA has carried out a study of past landing accidents and derived approximate values for the probabilities that given types of accident will be fatal. These values have been used in Safety Assessments prepared to date and are listed at Appendix 1.

Another mitigating factor for which credit may often be taken in an analysis is pilot action. The pilot may take over control to complete the landing manually or to abandon it. Any assumptions as to his ability to intervene effectively must take full account of the way he is alerted to the need to do so, and the situation at the time (flight path, trim, etc.). For example if the automatic landing system were to cause the aeroplane to land at or near the edge of the runway, it is accepted that the pilot can recognise this easily and take over when the system is being used in good visibility, but that in limiting visibility conditions he cannot. Therefore a system which is to be used in limiting conditions must be shown to be proof against this type of occurrence (or to provide an early and effective warning in the flight deck).

Probably the most difficult objective to frame is that relating to autopilot cut-out or loss of function particularly at low height in the very lowest visibilities. If this occurs immediately prior to touchdown with the flare virtually completed, the pilot might elect to continue to a landing under manual control, but if it occurs earlier in the approach or flare he would have to abort the landing and make a go-around. In either case it is difficult to estimate the risk involved.

### 6.2. Analysis Tasks and Division of Risk

Any particular hazardous effect may arise as a result of:

- (a) a single failure or combination of failures:  
the safety assessment should therefore contain a failure analysis.
- (b) the performance of a failure-free system due to tolerances and environmental conditions:  
the safety assessment should therefore contain a performance analysis.
- (c) a combination of failures and performance:  
the safety assessment may need to examine the performance of the system in some failure modes.

It may be helpful at this stage to take an example illustrating the framework into which the performance and failure analyses must fit. If we consider the case of a landing in which the vertical velocity at touchdown exceeds the design ultimate vertical velocity for the aeroplane, and if we suppose that one thirtieth of the total  $10^{-7}$  risk may be allocated to this cause then it must be shown that the probability of a catastrophe due to this cause is  $\frac{10^{-7}}{30} = 3 \times 10^{-9}$ .

We may now further suppose that a catastrophe is equally likely to arise from:

- (a) performance without any failure,
- (b) failures without any performance variability,
- (c) failures and performance, e.g. combining relatively probable failures with adverse environmental conditions.

Each of these categories can then contribute  $10^{-9}$  to the total probability of a catastrophically hard landing.

In this particular case pilot intervention does not provide any alleviation since CAA experience is that pilots do not reliably detect from visual cues that an automatic landing flare is about to result in a hard landing. In fact, if the system fails to flare the aeroplane the pilot is unlikely to intervene early enough to correct the situation unless he is given a positive warning to do so.



On the other hand a landing exceeding the ultimate strength of the undercarriage may be regarded as having catastrophic consequences on one in ten occasions (Appendix 1). Therefore, in the airworthiness objectives it will be apparent that a landing which is hard enough to break the aeroplane ( $>12$  fps) may have probabilities not exceeding:

- $10^{-8}$  per landing due to failures, e.g. failure to flare without warning,
- $10^{-8}$  per landing due to performance, e.g. the effect of an adverse gust,
- $10^{-8}$  per landing due to failures combining with performance.

For systems to be used in the lowest visibilities, both the failure analysis and the performance analysis must also assess the probability of loss of system function or "cut-out" (sometimes referred to as its "integrity"). CAA has accepted safety assessments which assume safe pilot take-over at the  $10^{-3}$  to  $10^{-4}$  level in these conditions. Therefore to meet an overall  $10^{-9}$  criterion the system cut out rate should be about  $10^{-5}$  taken over the last half minute or so of the flight. In practice this has proved to be extremely difficult to achieve and to demonstrate, even with fully fail operational systems.

The fact that some reliance can be placed on the pilot does ease the demands on the system and the analysis by comparison with say a fly-by-wire system where the safety of the aeroplane relies on continuing system operation. It will be a most challenging task to show that there is no sequence of failures or even a remote common mode of failure which will defeat the redundancy built into such a system to a probability of  $10^{-9}$  taken over the whole flight.

## 7. ANALYSIS

### 7.1. Failure Analysis

The analysis of possible failures of the system must start from the airworthiness objectives which establish the effects that are critical or hazardous. It is then a reasonable task to determine the condition of the system required to produce these effects, and the design principles which should be adopted to minimise the likelihood of their occurrence. In particular the design principles should define the precautions taken to limit the effects of failures (e.g. sub-system segregation, authority limiters) and the extent to which it is necessary to provide for continued operation after failures.

The analysis should take the form of a logical examination of all the individual failures, and combinations of failures, which can lead to the condition of the system under consideration. It should also contain a statistical evaluation of the probabilities of the failures to show compliance with the objectives.

In carrying out the statistical part of the analysis logical and careful thought must be given to the time periods which are significant to the particular probability being computed. For example a failure may only be able to cause a hazard during the last half minute or so of the flight. However, that failure may be dormant up to this point and may only declare itself during the landing. Thus in evaluating the probability that the failure will be present during the landing full account must be taken of the whole period back to the last point at which the system element was checked and found to be healthy. Consequently, an important product of the failure analysis is the maintenance procedure for the system since it must show that the checks are sufficiently comprehensive, and sufficiently frequent for system safety. To illustrate the point in a very simplified way, suppose that for a particular system the mean time between failures in the flare computation which lead to a hard landing is about 10,000 hours. If this part of the system is checked automatically at the start of the approach then the risk period over which it can fail is of the order of two minutes, i.e. the probability of a landing with this failure would be approximately  $\frac{2}{60 \times 10,000} = 33 \times 10^{-6}$  which does not meet the  $10^{-8}$  target.

There would therefore be a need for a second channel to "monitor" the first so that in the event of a failure, the disagreement between channels would cause the system to disengage and, provided the pilot is given an unmistakable warning, he can take over and make a safe landing. The probability of both channels failing following the approach check is negligible (approximately  $10^{-11}$ ), but of course that is not the end of the matter. It is also necessary to ensure that the approach check sequence itself can be relied on. This will almost certainly require that a ground check be carried out at some regular interval.

In general the failure analysis is concerned with satisfying the average risk criterion, and it is only occasionally that there is a need to consider the specific risk for the particular flight. That only arises when there is a question of using a system with a known failure.

The following gives some specific guidance on the detailed conduct of the failure analysis for a safety assessment.

- (a) The system which is being analysed must be precisely defined (boundaries, interfaces, modification standard etc.).
- (b) Where the effect of a failure is not readily apparent, either the most adverse possible consequence should be assumed, or such testing should be carried out as may be required to establish the effect without doubt.



- (c) A single failure may only be assessed as having a probability less than  $10^{-5}$  based on applicable service experience and analysis or alternatively a detailed engineering evaluation backed by testing.
- (d) A single failure may only be assessed as having a probability less than  $10^{-9}$  when it applies to a particular mode of failure (e.g. mechanical jamming) and it can be shown that such a failure need not be considered as a practical possibility.
- (e) In systems which rely for their airworthiness on redundancy, particular attention should be paid to common mode failures i.e. multiple failures arising from a single cause. Such a common cause might be:
  - local fire
  - electromagnetic interference or electrical transient
  - mechanical vibration
  - leakage of water
  - failures of cooling systems.
- (f) The influence of other systems should be taken into account.
- (g) When the failure of a device can remain undetected in normal operation, the frequency with which the device is checked will directly influence the probability that such a failure is present on a particular occasion.
- (h) When the failure can be expected to result in other failures, then account should be taken in the analysis of these further failures.
- (i) The features which are shown by the analysis to be critical should be reviewed to determine whether modification action should be taken.

## 7.2. Performance Analysis

Many different factors affect the performance of a system, and to some remote probability a number of them may combine together in an adverse sense sufficiently to cause an accident. For an automatic landing system, it is the factors external to the system which have a dominant effect on the performance rather than tolerances. These are generally of minor significance but may not be negligible. The factors which matter will depend on the performance parameter being considered. Wind shears and gusts will obviously influence lateral as well as vertical flight path control, but noise or bends on the ILS localiser beam will only affect azimuth control. The flare-out manoeuvre and the touchdown impact are mainly influenced by wind shear and gusts, and by the profile of the ground immediately prior to, and during the landing. The ground profile effect arises because the main terms in the control law are based on the height of the aeroplane wheels above the ground as measured by a radio altimeter, and the flare computation is therefore sensitive to ramps, steps or bumps on the ground. However, this is generally not a severe problem because the flare manoeuvre is largely carried out when the aeroplane is over the paved runway surface. Noise, or "bends", on the ILS glide slope beam may also contribute, but generally only to a small extent.

The method which has been adopted for performance analysis consists in establishing the statistical distribution of the parameter being considered, and then determining from that the probability that it will reach a hazardous level. For example considering the hard landing case again, the procedure would be to establish the distribution of vertical velocity at touchdown, and then to determine that it will not reach a value exceeding the design ultimate value (normally 12ft. per second at maximum landing weight) with a probability greater than the desired value of  $10^{-8}$ .

Some of the factors which influence performance are "deterministic" in nature. For example, all other things being equal, the touchdown point along the runway may vary with the centre of gravity of the aeroplane. But the c.g. itself will have a statistical distribution so that the overall outcome is a statistical distribution of touchdown position as affected by centre of gravity. Other factors such as turbulence, or beam noise, are themselves essentially statistical and will also generate a distribution of touchdown points, etc. All these individual distributions combine statistically to generate the overall statistical distribution for the parameter in question. (For gaussian distributions with the same mean, the variance of the overall distribution is the sum of the individual variances.)

The problem of course is to establish the statistical distributions of the critical parameters. Clearly flight testing is necessary, but limited by the time and money available, and certainly not to anything like the levels necessary to "prove" probabilities in the order of  $10^{-8}$ . Extensive use is therefore made of simulations with statistical models of the distributions of the disturbing parameters, simulation of the aeroplane and, either simulation of the automatic landing system or real hardware units. Whereas flight test landings will be numbered in hundreds at most, the simulation runs may run to thousands or hundreds of thousands. This is still regrettably small by comparison with  $10^{-8}$ , so that the statistical distributions which have been measured must be extrapolated before reaching conclusions (inferences) regarding safety. The confidence with which this can be done is completely dependent on the quality and quantity of information which has gone into the modelling of the critical variables, and the accuracy of the aeroplane and system simulation. A clear understanding of the system and the factors which are most critical to it will enable the designer to concentrate his attention on the effects of the more extreme conditions, i.e. the "tails" of the distributions.

Although statistics gained from flight testing can not be used on their own to establish the performance of the system, it is of course a prime requirement that distributions derived from flight testing should be compatible with and support the simulation work. Without that, there can be no confidence in the conclusions of the analysis.

An important product of the performance analysis is a statement of the limitations to be observed in the use of the system, e.g.

- the quality of the ILS localiser or glide slope
- wind speed (cross, tail or total)
- any limitation on aeroplane weight or centre of gravity, additional to those applied to manual landing.

These are determined as the maximum value of the particular parameter which will permit compliance with the average risk criterion, or when appropriate, with the specific risk for the particular flight.

### 7.3 Failures and Performance

In describing the analysis of failures (para. 7.1) consideration was only given to those which are certain to be catastrophic, and those which have a high probability of being so (e.g. a failure to flare at 1 : 10.).

However, there is a class of failures which have less severe consequences and may therefore be permitted to occur with a higher probability. For example a failure which leads to a partial flare may leave the system able to make quite satisfactory landings except in the presence of the more extreme gusts. For example if a failure of this nature had a probability of  $10^{-4}$  per landing, then it would be necessary to analyse the performance of the system in its failed state to determine that the probability of a hard landing is less than  $10^{-4}$ .

Great care is needed to ensure that this does not add enormously to the amount of performance analysis to be carried out. In general pessimistic simplifying assumptions should be used to clear all but the most marginal of the cases which have to be considered.

### 7.4. In-Service Proving

It has become the practice in the United Kingdom that automatic landing systems are subjected to a comprehensive scrutiny in day-to-day operations before they are released for use in the lowest visibilities. Formal certification for low visibility operation of the system is withheld until a period of in-service proving has been completed. Analysis of pilot reports and on-board flight data recordings is used to monitor the performance, integrity and failures of the system in line service. There are two main channels of activity - "routine" and "special events".

Analysis of a large batch of landings is used to confirm that the performance of the system on a day-to-day basis, and that the reliability of the equipments is compatible with the conclusions of the safety assessment.

Any landings which appear to fall outside the expected range of performance variations, and any failure effects which appear to violate the safety assessment are treated as "special events" and are subjected to specific and detailed examination. In many cases the incident is resolved satisfactorily, but in a small number of cases the result of the investigation is a hardware modification, a changed procedure or some amendment to the limitations on the use of the system, and none of the systems proved in-service up to date has escaped such action. There is no doubt that this period of in-service proving provides an essential back-up to the safety assessment procedure for automatic landing systems which are to be used in "blind" conditions.

## 8. OTHER SYSTEMS

General airworthiness requirements for systems also contain explicit statements of acceptable probabilities for failures, expressed in words rather than numbers and categorised according to the severity of their effect (Extremely Improbable, Extremely Remote, Remote, Probable, etc.). Acceptable numerical interpretations have been specified and have been used as the basis for many safety assessments over the past five years or so.

The CAA is consulting with industry on draft proposals which would clarify and amplify these requirements. In essence these proposals provide for:-

- (a) design review of a new aeroplane type following which there would be agreement between CAA and the manufacturer on the systems which require a safety assessment procedure, and those which can be accepted against conventional requirements,
- (b) airworthiness objectives to be agreed for each safety assessment as a list of effects associated with the system under consideration, and the allowable probability,
- (c) analyses of failures and their effects, including combinations of failures,

- (d) statistical analysis only to be carried out where the conclusions of the failure analysis do not show obvious compliance with the airworthiness objectives, and especially in the case of complex systems, new technology, etc.,
- (e) performance analysis of those systems where performance variability may have critical consequences,
- (f) safety assessment documentation adequate to ensure that throughout the life of the aeroplane modifications to the system can be designed and implemented in such a way that the airworthiness objectives continue to be complied with.

#### 9. MANAGEMENT

Finally, it is worth dwelling briefly on the management of a safety assessment programme. Its most important contribution to safety is that the system concerned is subjected to a systematic and critical analysis. It is necessary that the people who do this task understand the system and its functions, and the airworthiness objectives. They should be engineers with a knowledge of the system, not statistical mathematicians or computer programmers. It is the analysis of the system which is important in the safety assessment, not the arithmetic. Ideally the safety assessment team should be a group of people separate from the design force and particularly so in terms of management. As far as possible they should not be responsible to the managers of the design team.

Of course, the design team should also be involved since they must know and work to the disciplines and constraints imposed by the safety assessment. System "strategy" or "architecture" can have a great influence on the size and complexity of the safety assessment task, so that where possible the designer should always consider the analysis and certification programme in making design decisions.

Finally there can be no doubt that the top management of the design organization must understand and support the safety assessment programme, and must be prepared to implement any actions it calls for.

#### 10. CONCLUSION

Automatic landing systems and their use in low visibility must be subject to a systematic safety assessment programme to ensure an adequate design standard to determine the limitations within which the system may safely be operated, together with the necessary maintenance and operating procedures. This technique can be applied to other complex or novel systems.

#### 11. REFERENCE

1. "Airworthiness Requirements for Automatic Landing including Automatic Landing in Restricted Visibility down to Category 3", British Civil Airworthiness Requirements Paper No. 367, June 1970.

APPENDIX 1

## THE RISK OF FATALITY ASSOCIATED WITH A LANDING INCIDENT

It is recognised that many incidents may not necessarily result in fatal accidents. Therefore, in assessing the overall safety of the operation the probability of an incident may be weighted according to its fatality risk. The following is a list of incidents and values for the fatality risk in each case.

These values are considered to be representative of past experience, and are normally accepted if used in the estimation of risks in a safety assessment. However, if it were clear that some new aeroplane was substantially less crashworthy in any or all of the incidents listed, the fatality risks would have to be revised for that aeroplane and those incidents.

<u>Incident</u>	<u>Fatality Risks</u>
(a) Aeroplane leaves the airspace which is guaranteed to be free of obstacles in either approach or go-around, or, if the excursion is small and of short duration.	fatal 1 : 30
(b) Aeroplane touches down short of the runway, but not more than 200' short.	1 : 30
(c) Aeroplane touches down more than 200' short of runway.	fatal
(d) Aeroplane touches down to side of runway but with wheels not more than 250' from centre line.	1 : 30
(e) Aeroplane touches down to side of runway, but wheels more than 250' from centre line.	fatal
(f) Aeroplane runs off side of runway but wheels remain within 250' of centre line.	1 : 30
(g) Aeroplane runs off side of runway with wheels more than 250' from centre line.	fatal
(h) Aeroplane runs off end of runway, not more than 200' from runway.	1 : 100
(i) Aeroplane runs off end of runway to more than 200' from runway.	fatal
(j) Aeroplane touches down harder than design ultimate vertical velocity.	1 : 10
(k) Aeroplane touches down with sufficient lateral tracking velocity or yaw to collapse undercarriage.	1 : 10
(l) Nosewheel or rear fuselage touch ground before main wheels.	non-fatal
(m) Pod or propeller touch ground.	non-fatal
(n) Wing tip touches ground after undercarriage.	non-fatal
(o) Wing tip strikes ground before undercarriage.	fatal
(p) Aeroplane stalls.	fatal
(q) After initiating go-around aeroplane strikes the runway.	likely to vary from aircraft to aircraft

The incidents defined above assumed that the airfield meets the recommendations of ICAO Annex 14 with regard to the Strip surrounding the runway. This extends 200' before the threshold and to 500' from the centre line on either side. Of this only the first 250' either side has a prepared surface, and the outer portions of the Strip are merely cleared of obstructions. Where these recommendations are not met, the incidents would need to be re-defined, and, for example, if waiting aeroplanes or ground vehicles were permitted on the outer part of the Strip, the incidents reflecting lateral displacement would need to be in terms of wing-tip and not wheel displacement.



## FUTURE TRENDS IN HIGHLY RELIABLE SYSTEMS

James I. Arnold  
THE BOEING COMPANY  
3801 South Oliver  
Wichita, Kansas 67210

### SUMMARY

The need for highly reliable flight control systems in both control configured vehicles (CCV) and conventionally designed aircraft is discussed. Technology trends in the area of control system computation, electronics, sensors and actuation are addressed. Increased use of digital computation and signal multiplexing in future control systems is considered inevitable. Recent technology developments in high density electronic packaging, large scale integration and fiber optics will be applied to achieve highly reliable electronic systems. Component designs will be required to withstand potentially severe environments in the presence of lightning or nuclear phenomena. Redundancy management will continue to be a prime driving force in reliable system designs. The use of in-line monitoring to limit the proliferation of redundant channels should find application in future systems. Maintenance and preflight self-test systems will play an increasingly vital role in assuring the integrity of redundant flight-critical systems.

### 1.0 INTRODUCTION

Current trends in aircraft research and development indicate that future aircraft will rely more heavily on flight control systems to provide improved performance and more economical flight operations. Automatic control systems will assume new pilot assist roles, and will be employed in the emerging active control technology concepts. These systems will be critical for flight safety in some or all of the aircraft flight envelope. Implementation needs that result from these trends are increased system reliability, lower cost, and size and weight improvements.

The required reliability will be attained through development of higher reliable components and better redundancy management techniques and using built-in test methods to monitor system status. The future flight control systems will rely more on digital techniques and less on analog implementation methods to provide the required reliability. These systems will employ fly-by-wire implementation, because of the increased versatility attainable, and will incorporate fiber optic techniques because of the increased mission reliability and redundancy that is possible.

Significant advances have been made, and will continue to be made, in computer parts technology. Despite dramatic improvements in digital technology, further reductions in cost and size will be rather modest because the digital portion of a digital flight control system represents approximately 35 percent of the required functions. Analog electronics, power supplies, wiring, etc., form the balance. Thus, further improvements are dependent upon analog technology advances which cannot be predicted as well as advances in digital technology.

Improvements in sensor technology have also been lacking during the last twenty years, making it difficult to predict future technology advances. Rate and acceleration sensors used in aircraft of the 1950's are essentially the same as units being installed in the F-16 fighter. Current research in sensor technology is directed toward improved reliability. The concepts being considered for rate sensors eliminate the rate gyro spin motor and inherent bearing failures. Redundant skewed sensor packages with redundant computers to process the data for all aircraft subsystems are also being investigated. Some of these concepts are certain to be employed in highly reliable automatic flight control systems of the future.

Highly reliable flight control systems on future aircraft will use redundant actuators driven by high pressure hydraulics to reduce component weight and space. The use of fly-by-wire will permit elimination of primary control linkages, rods and cables. Redundancy configurations will be tailored to the individual aircraft requirements. Those being considered include tandem cylinders, multiple cylinders arranged side-by-side along the surface hinge line, and combinations of parallel, independent surfaces operated by individual, or multicylinder, actuators.

The following sections discuss future applications of highly reliable control systems and technology trends in control system computation, electronics, sensors, and actuation systems to meet these application needs.

### 2.0 BACKGROUND

In the future, increased dependence will be placed on flight control systems for flight phase critical functions, such as automatic landing and guidance, and aircraft configuration dependent functions, such as active control systems and fly-by-wire (FBW). Systems falling into the first category are those used on aircraft that are basically stable throughout the flight envelope without the system operating, but the short term effect of system failure in certain flight phases can result in loss of the aircraft. Those falling into the second category are systems used on aircraft which rely on the systems to meet

basic flutter, stability, or load design requirements during some or all of the flight envelope, and system failure would lead to loss of the aircraft if it occurs in a critical flight condition.

Safety of flight requirements demand improvements in system reliability. This can be achieved through a combination of improved component reliability and through extensive and effective redundancy to achieve fault tolerance.

## 2.1 Flight Phase Critical

Flight control systems used for flight phase critical functions usually perform a pilot assist role; that is, the flight crew could perform the same functions, but an automatic control system can perform the task more precisely and with greater safety. A primary example of such systems is the automatic landing systems employed on the Boeing 747 and other commercial aircraft.

The 747 automatic landing system includes the Sperry Rand SPZ-1 autopilot-flight director system and provides fail-operate capability (Reference 1). This system has the capability of sustaining a failure while automatically guiding the aircraft to touchdown using existing instrument landing system ground radio facilities. The performance, operational procedures, safety and reliability of this equipment are consistent with requirements for operation under Category IIIB weather conditions. Fail-operational autoland will be the basis for progress toward automatic landing capability under the more severe Categories IIIB and IIIC weather conditions. To meet the requirements imposed by these categories, improved ground-based radio references and an independent visual display system for the flight crew seem essential. But, the most significant requisites for implementation of all-weather landing capability are further experience with current autoland techniques, continued equipment enhancement, and closer cooperation between manufacturers, airlines, pilots, and regulatory agencies.

The National Aeronautics and Space Administration is conducting a long-term program called the Terminal Configured Vehicle, or TCV, program, to address some of the major problems involving transport aircraft operating in terminal areas calling for new or improved capabilities in airborne systems (Reference 2). This flight research program uses a Boeing 737 modified to incorporate an aft flight deck and advanced on-board electronic systems. The aft flight deck is a second cockpit, installed in the passenger cabin, which will simulate a normal flight deck. The two man crew in the aft flight deck are able to fly the airplane from takeoff through landing.

The advanced electronic systems include a triply redundant digital automatic flight control system and a digital navigation guidance and display system, including cathode ray tube displays. The systems integrated into the 737 airplane enable inflight experiments supporting research into terminal area operations, including noise abatement approaches, precision 3-dimensional area navigation, and time navigation to reduce delays and fuel expenditures and to increase airport capacity and landing operations in conditions down to Category III. These experiments will lead to definition of needed aerodynamic and avionic features necessary to operate in future high-density terminal areas.

## 2.2 Aircraft Configuration Dependent

Fly-by-wire and active control technologies have progressed to the point that they can be applied to prototype and production aircraft (Reference 3). Analytical studies and flight demonstrations have generally proven the advantages of FBW and active control systems in terms of aircraft performance.

Fly-by-wire increases the number and complexity of control functions that can be easily implemented. Implementation of active control technology concepts does not depend on the adoption of fly-by-wire, but the greatest payoff for both is realized when they are integrated. An airplane designed for maximum utilization of active control technology concepts will most likely use FBW in the system implementation because of the flexibility available to the designer, such as in bringing together signals to a common control surface from several sources in a straight forward manner.

Active control technology offers a new aircraft design philosophy that provides increased design freedom. The active control functions that provide the most potential for improvement are:

- Flutter Mode Control
- Maneuver Load Control
- Ride Control
- Gust Load Alleviation
- Augmented Stability
- Fatigue Reduction

All of these concepts except gust load alleviation, were successfully flight demonstrated during the B-52 Controls Configured Vehicle program. In addition, all of these concepts except flutter mode control have been individually committed to production to some extent. A ride control system is being used on the 747 airplane and the B-1. The F-16 selected by the Air Force for production uses augmented stability. A combined fatigue reduction and maneuver load control system is being installed on the C-5A to extend the wing fatigue life.

The full potential of active control technology can be achieved only by incorporating the concepts in the initial design phase of a new airplane. This approach involves integrating proven flight control technology into the aircraft configuration definition on an equal basis with the design technologies of aerodynamics, structures and propulsion, as shown in Figure 1. Aircraft configurations which integrate active controls in the design, are dependent on the control system to meet design objectives. For such an aircraft the control system must be as reliable as structure to have a configuration competitive with conventional aircraft.

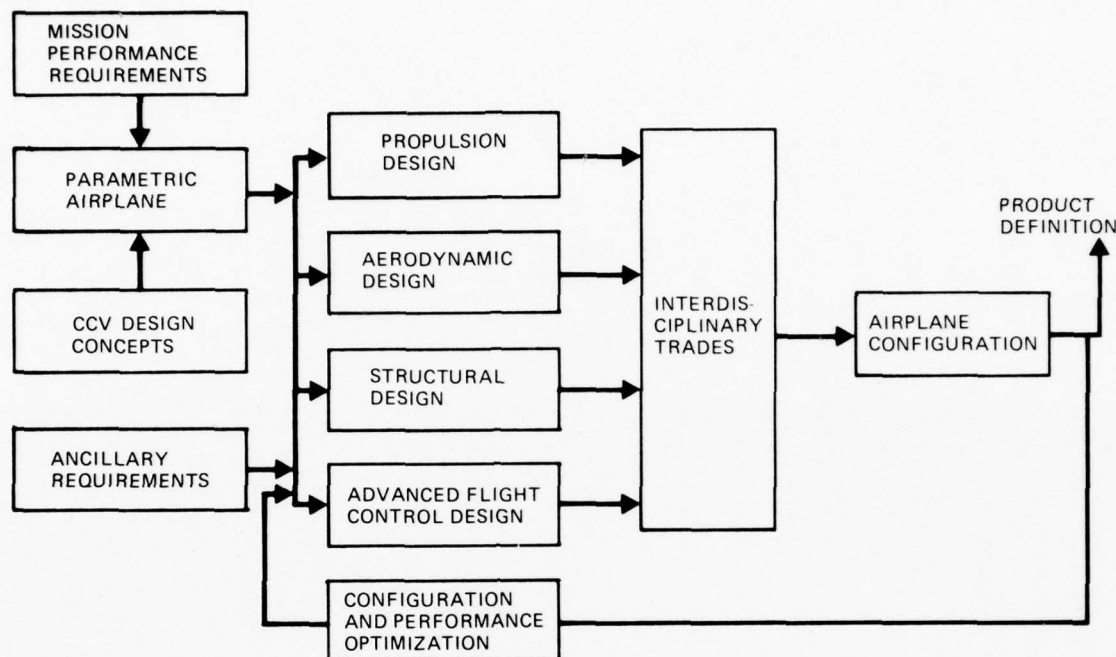


Figure 1. Active Control Technology Airplane Design Cycle

Future aircraft designs being considered at the present time which will require active control technology concepts are discussed briefly in the following paragraphs. These examples illustrate the increasing complexity of future aircraft control systems and resulting need for higher reliable systems.

A transport with asymmetric wing, being considered by NASA, may offer performance potential sufficient to bring about its eventual development (Reference 4). The fuselage of this aircraft, designed to cruise at high transonic speeds, would be long and slender without the conventional stiffening provided by the wing root structure. It is probable that a ride control system would be necessary to provide a passenger comfort level commensurate with current transport aircraft. The motion character of the airplane is somewhat unusual, principally because of aerodynamic coupling. A stability augmentation system would be required to decouple the motion and provide capability for conventional piloting techniques.

The freighter concept shown in Figure 2 would make extensive use of active control concepts to achieve its maximum performance potential (Reference 5). The airplane is designed such that empty weight is only 25 percent of the gross weight, compared to about 40 percent for current technology freighter designs.

Trim and maneuver load control systems would permit adjustment of the airdload distribution to closely match the weight distribution, thus greatly reducing wing bending moments and leading to reduced structural weight. Additional performance improvement could be obtained by utilizing augmented stability and positioning the center of gravity for maximum performance. This airplane will probably require digital control system technology, with its capability for monitoring system state and selecting healthy channels. The size of the airplane, together with the large number of separate control surfaces and the necessity for individual commands, leads to a natural use for fly-by-wire.

Another application of active control technology demonstrating the complexity possible is the heavy-lift airship concept proposed by Goodyear (Reference 6). Figure 3 shows a model demonstrating the concept, where four Sikorsky CH-54B helicopters are attached to a 2.5 million cubic foot Dacron/neoprene hull. The four helicopters would be controlled in parallel from a single cockpit by a digital fly-by-wire control system.

The helicopters are gimballed in pitch by main rotor cyclic pitch and driven by servo controlled actuators in roll to offset gimbal coupling forces resulting from rotor torque. The tail rotors on the aft helicopters are replaced with propellers and reoriented to provide pusher capability required for forward flight in the unloaded condition. The airship would be flown using standard helicopter controls. All helicopters would be controlled by a command pilot who would have electric cyclic and collective sticks and rudder pedals that generate fly-by-wire commands to all four helicopters. A fly-by-wire system for controlling a tandem helicopter was developed by Boeing-Vertol during the recent heavy lift helicopter program.



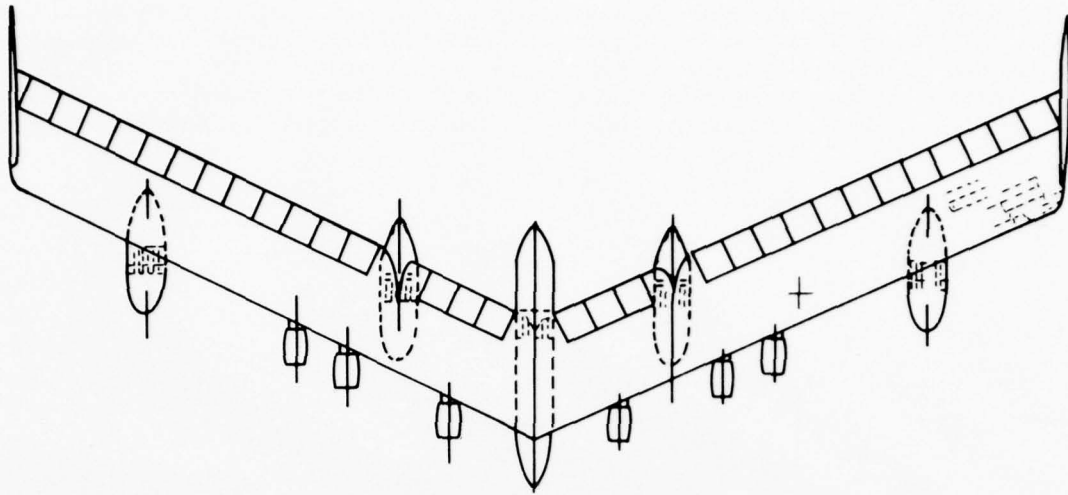


Figure 2. Special Purpose Freighter



Figure 3. Goodyear Heavy-Lift Airship Concept

The airship would be taken off and landed vertically under manual control of the command pilot. Manual controls for longitudinal and lateral flight, hover, and pitch and roll trim would also be provided. The automatic flight control system would include heading, altitude and pitch attitude hold modes for forward flight. The precision hovering system would have longitudinal and lateral position holds with creep capability altitude hold, and roll and pitch attitude hold.

To meet future demands for improved control system reliability, significant advancements will be achieved in system components, redundancy management, and built-in-test techniques. The following sections summarize some of the more significant trends.



### 3.0 COMPONENT RELIABILITY TRENDS

#### 3.1 Computational Trends

In the computational area, a definite trend toward digital implementation is seen in both commercial and military aircraft. Digital techniques which have already been applied to various subsystems on current wide-body transports, are now being evaluated for the more critical flight control system applications. The expected advantages such as reduced initial costs, improved reliability, easier trouble shooting and greater flexibility are generating considerable interest in the digital approach. Boeing and McDonnell Douglas are already studying digital autopilot mechanizations for new transports known as 7X7 and DC-X (Reference 7). It has been estimated that a digital mechanization of the L-1011 autopilot could result in savings of 25 percent in initial cost, 30 percent in maintenance costs and 20 percent in weight. In addition, it should provide a 25 percent improvement in reliability.

While digital flight controls are generally most attractive for new aircraft where they can be designed integrally with the airframe without an unusual nonrecurring expense burden, some retrofit interest is seen. McDonnell Douglas, for example, has included a digital flight control system in one of its DC-9-50 aircraft proposals. Boeing is also studying digital control retrofits for several of its products, including the 727 and B-52 aircraft. These systems would combine existing sensors and subsystems with a new digital computer that includes a very high degree of fault detection capability.

##### 3.1.1 Digital Versus Analog Implementation

The digital computer has several potential advantages for reliable control system applications. First, it can provide superior test coverage with less hardware than equivalent analog built-in-test (BIT) systems. A typical analog BIT with 85 to 95 percent test coverage requires bit circuitry that amounts to 20 or 25 percent of the total system hardware (Reference 8). In an equivalent triplex digital mechanization, the additional memory and I/O hardware requirement would comprise only one to four percent of the total system memory and interface. Second, a digital mechanization eliminates tolerance accumulation. Third, sophisticated signal selection algorithms, reasonableness testing and performance monitoring are possible which far exceeds analog system capabilities. Fourth, serial intercomputer links can be used to reduce the volume of cross channel wiring.

The digital computer does have some potential disadvantages, with respect to reliable systems, however. Failure modes and effects tend to be difficult to characterize and some failures tend to be difficult to detect with software self-test alone. In addition, digital computer implementations are susceptible to multiple channel generic software failures. Solutions do exist for these limitations, therefore they should not significantly impede the trend toward digital computation. For example, a low cost microprocessor monitor could be added to each channel to detect both computer hardware and software errors.

##### 3.1.2 Implementation Technology Trends

There are a number of current developments and trends in digital system implementation technology that will contribute to the advancement of reliable flight control systems. This section will concentrate on computational subsystem trends; component trends will be discussed in subsequent sections.

The trend in overall airborne computational system architecture is toward distributed systems, which simply means that each aircraft system function, such as flight control, has its own dedicated processors, but is able to exchange data with other systems, such as navigation and air data (Reference 9). Factors influencing the trend include component cost and conflicting redundancy requirements of various subsystems. The continually reducing cost of large scale integrated circuit computing components is reducing the economic advantage of the large central computer. In addition, the requirement for flight critical functions to be redundant tends to militate against mixing them with less critical functions. To do so would impose the same rigid level of testing and control on noncritical software as for critical system software. This would result in an unnecessary cost penalty in most cases.

State-of-the-art developments in the area of large scale integrated microcircuits have made it possible to consider the use of multiplex communications links in future flight control systems. Flight control systems and most particularly, redundant systems generally require large numbers of communications paths between the flight control computer and various subsystems which interface with it. These subsystems include sensors, actuators, controls and displays, in addition to other computer based subsystems, such as navigation. The potential benefits of multiplexing include reduced wiring complexity and weight, interface standardization, system flexibility (can add sensors without extensive rewiring), reduction in connector pins needed and improvement in electromagnetic compatibility due to fewer radiating wires.

Future reliable flight control systems will utilize optoelectronics and fiber optics for interchannel communications. Optoisolators can be used to eliminate ground loop problems while fiber optics provide electromagnetic interference-free cross digital communications between channels and subsystems.

A definite trend toward asynchronous operation of digital computers in redundant channels is seen. Flight experience with redundant digital systems, such as the Air Force Flight Dynamics Laboratories A-7 tests and simulation studies have pointed out the hazards of synchronous computer operation. One such study was conducted by Lear Siegler.

The purpose of the Lear Siegler simulation study was to examine input accuracy and bias effects, independent sampling/computation rates, high input signal rates, and integrator divergence.

In the study of the trade offs between synchronous and asynchronous operation, those factors which would appear to be differences between the two approaches are discussed (Reference 10). The main points answered are the benefits which are offered in either approach, the mechanization differences, the operational differences and the reliability differences. This study verifies that asynchronous operation provides more reliability when employed in redundant digital flight control systems through the avoidance of the potential single point failures of synchronized clock operation. It also avoids the necessity to develop and qualify redundant hardware clocks or to develop and validate the software logic for a software clock. This will result in lower program costs for the asynchronous approach. Another benefit of asynchronous operation is the reduction in EMI induced control surface transients. This results from the fact that there is a low probability of all channels being in the same computation cycle at the time the EMI effect is present. The subsequent output signal selection will eliminate the single channel transient.

A trend towards using a library of software modules which can be tailored and linked to fit the software requirements of a specific system will have important impacts on both system cost and reliability (Reference 9). This is a significant change from current avionics software practice in which ad hoc techniques are used on a system basis, producing software that is both expensive and unique. Being able to select and tailor already validated modules to satisfy a new requirement also contributes to reliability because validation and verification of the software will tend to be more complete.

A trend towards the use of higher order languages is an important companion of the library of modules trend in order for the library to be transferable from one computer to another. Structured programming and software testing techniques will tend to alleviate the reliability questions sometimes associated with computer object code generated by higher order language compilers.

Technology advancements in areas of microprocessors and memories are destined to produce a large impact on the development of future reliable control systems. Mass memories, now dominated by rotating disks and tapes, will soon be replaced by solid-state equivalents, such as magnetic bubble and charge-coupled-device memories. Core memories and volatile semiconductor memories will undoubtedly give way to nonvolatile semiconductor memories which are electrically alterable.

### 3.2 Electronic Trends

The control system technology area which is currently undergoing the most rapid development is electronics. Growth areas of interest to reliable system designs include large scale integrated circuit technology, microprocessors, memories, data converters, packaging technology, mass storage and fiber optics.

Probably the biggest single contributor to the current downward trend in the cost/performance ratio in digital electronics can be attributed to advancements in large scale integrated circuit production (Reference 9). This technology allows hundreds, or even thousands, of logic functions to be implemented on a single integrated circuit chip. Because the use of these chips reduces parts count, reliability as well as cost is improved. This technology has led to low-cost off-the-shelf microprocessors, memories and various other computational building blocks which can be used in flight control systems.

The dramatic growth of integrated circuit production densities in the seventies is illustrated in Figure 4. Since 1971 memory densities have increased by more than an order of magnitude; from one thousand to sixteen thousand bits per chip. Logical integrated circuit densities have made similar gains. It is not unreasonable to predict that production densities will reach sixty thousand by 1980.

The LSI technology has spawned a significant new technology called microprocessors. The microprocessor is defined as a standard programmable LSI which consists of a parallel arithmetic unit, a control unit, and a general purpose parallel data bus for memory and external device communications. This chip (or chip set) can be combined with LSI memory chips to realize a general purpose microcomputer for extremely low cost.

These microcomputers, having the programmability of conventional general purpose computers, will be used to perform computational functions and to replace hard-wired logic. In both applications a significant fact is that there is no longer a driving force to use the device efficiently. System level cost trade offs tend to lead to dedicated use of these devices for certain functions even though this may result in, for example, the device being kept busy only 20 percent of the time.

In random access memories, LSI technologies have already given a clear indication that they are replacing expensive and space consuming core memories (Reference 11). In some current designs, read only memories are being used for program instruction storage while volatile semiconductor random access memories are used for variable storage. The trend is toward eliminating core even for critical flight control applications where nonvolatile variable storage memory is a requirement. Fast nonvolatile semiconductor read/write memories will soon become available to fill that need. Also noteworthy is the development of charge-coupled device (CCD) memory systems as solid-state replacements for disks and drums. In architecture, the CCD memory differs from the familiar ROM or RAM so radically that it deserves further discussion. The new devices basically simulate the operation of rotating drums. For example, one 16,384-by-1-bit chip is organized so that it can

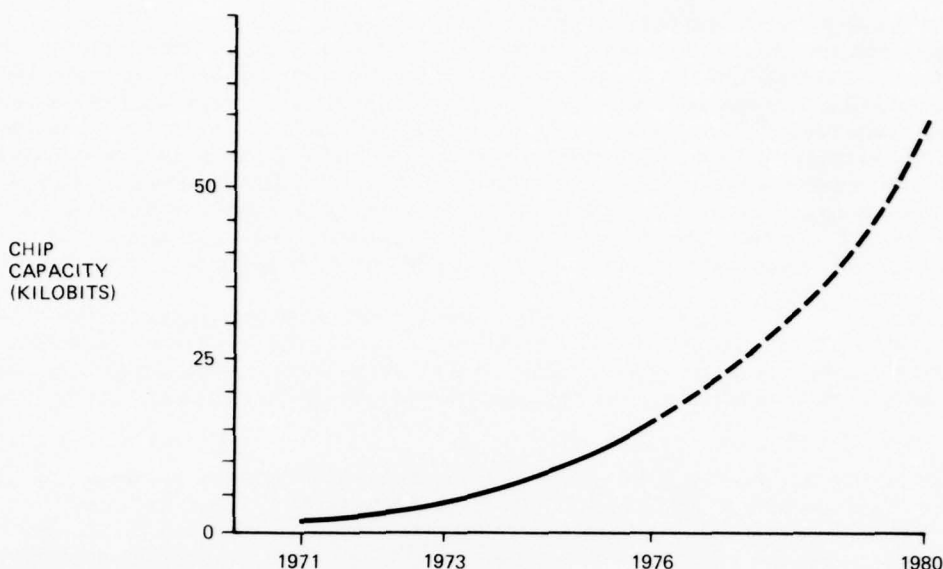


Figure 4. Integrated Circuit Production Densities in the Seventies

combine serial and random-access functions. The devices are arranged as 64 256-bit shift registers in which four-phase clock signals simultaneously shift the data. Each shift register can be thought of as representing a single track in a conventional drum, and each track can be thought of as being divided into 256 sectors corresponding to the 256 CCD data storage cells in each line. The "rate of rotation" of this semiconductor drum therefore is controlled by the four-phase clock.

CCD memories have clear advantages over existing high-speed drums. They are an order of magnitude faster, despite the fact that their data is accessible only by block or line. At present, any line is accessible in 100 microseconds, and that time is expected to decrease significantly over the next few years. CCD memories also have no need for high-speed mechanical movements and should therefore be much more reliable and last longer than today's drums with their extremely fast rotating mechanical assemblies. Estimates are that a simple five-board CCD system could be only half the cost of a 5 million-bit rotating system.

Additional electronic component advances of interest to reliable flight control designers include improved data converters, operational amplifiers, and optical couplers (Reference 12). Microcircuit size monolithic analog-to-digital and digital-to-analog converters which require no external components will become available. These are already available in hybrid models with somewhat larger dimensions, or in monolithic circuits which require a few external components. Improved field effect transistor input operational amplifiers are being developed using ion implantation production techniques. High gain optical couplers for both analog and digital applications are on the way. These will be used to avoid failure propagation paths for more reliable redundant systems.

New methods of packaging electronics will see more widespread application in reliable control systems (Reference 13). Techniques such as die-stamped circuit boards, multiwire, and stitch wiring will provide manufacturing alternatives that are free of environmental impact. Of the packaging techniques mentioned, stitch wiring is perhaps the most attractive for flight critical systems because of its very high reliability.

Hybrid circuits are emerging as an extremely high density method of packaging digital electronics. Thick-film hybrids normally are thought of as relatively simple analog circuits on relatively small substrates, like a quarter inch, but this is a misconception. Many thick-film hybrid manufacturers now are turning out large-scale digital hybrids in units as large as nine inches square and containing upwards of 220 monolithic integrated circuit chips. These units, aimed at the military/avionics/space field, are possible because of the development of multilayering in thick-film hybrids and the greater use of computer-aided design and manufacturing. Digital circuit densities as high as 40 monolithic IC chips per square inch are possible by this multilayered technique as compared to only 2 DIP IC's per square inch for conventional packaging.

Fiber optics will become more and more attractive to redundant system designers as optical fiber, connector, emitter and sensor technologies continue to develop (Reference 14). The primary reason for considering optical data transmission for fly-by-wire systems is its great potential for improved survivability to physical damage and electromagnetic phenomena such as lightning strikes. The survivability advantage over wire is achieved because an optical cable can be divided into two parts and recombined by means of Wye connections, with data transmission integrity maintained by only one of the paths. The second path may be damaged or destroyed without coupling a disruption into the remaining path. This is, of course, not possible with conventional wire where a short circuit in any strand causes total failure of all points tied to that wire strand. The potential for lightning survivability is estimated on the basis of immunity to induced currents in the optical cable and the electrical isolation of all circuitry from that cable.



To illustrate how an optical channel has a greater damage survivability potential than an electrical data channel for a fly-by-wire application, consider a quad redundant configuration of four control computers transmitting control data to control actuators. Now, as illustrated in Figure 5a, suppose we wished to improve damage survivability by running every wire twice, once along the left side and once along the right side of the aircraft. If damage on one side of the aircraft merely opens a wire, then we would indeed have four surviving data transmission paths, one for each of the four control computers. However, *damage is rarely so benign, and one should expect that the damaged conductors would be shorted to each other or to structure (ground) as shown in Figure 5a.* With such electrical shorting of conductors, the redundant set of wires on the undamaged side are of course useless. (A short of only one strand in a multistrand electrical conductor destroys the function of the entire conductor.) Consequently, wiring a quad redundant fly-by-wire system as shown in Figure 5a would be a fundamental violation of system safety criteria because a single source of damage could fail all four control channels.

The proper way to wire a quad redundant, electrical fly-by-wire system is illustrated in Figure 5b. Here, two channels are wired along the left side of the aircraft and two along the right side of the aircraft. Damage vulnerability is still severe as shown in the illustration, for now two cables and hence two of the four channels can be disabled by a single source of damage. Thus a double fail-operative fly-by-wire system can be reduced to a fail-passive, dual channel system following a single source of damage.

The advantage of the optical system is illustrated in Figure 5c. Each of the four computers provides one cable along the aircraft's left side and one along the right side. This is the same configuration as in Figure 5a except that an optical short cannot occur if a cable is destroyed. Hence, the left and right side transmissions are uncoupled. Destroy all the cables on the left side, and all four cables on the right side continue transmitting as if nothing had happened. The quad redundant system remains double fail-operative despite destruction of all cables on one side of the aircraft. This remarkable survivability can be achieved with fiber optics because the equivalent of an electrical short cannot occur in a fiber optic cable. That is, the only failure mode is the equivalent of the electrical open.

### 3.3 Sensors

The sensors most commonly employed in automatic flight control systems are gyros, accelerometers, and differential transformers. Reliability of linear variable differential transformers is well established, and they will continue to be used successfully on future aircraft. Servonulled linear accelerometers have usually proved more than adequate in most past applications, and improvements are being made. Probably the most significant improvements will be made in angular rate sensors.

Pendulous force-rebalance linear accelerometers will continue to dominate reliable control system applications. Typical units will have no wearout modes and will offer very high reliability. One example is the new Honeywell accelerometer that incorporates a unique mechanization resulting in low cost and high accuracy with time and environmental exposure (Reference 15). The pendulum and suspension are fabricated from quartz fibers arranged as shown in Figure 6.

A thin film of silver is vapor deposited over the quartz suspension and pendulum. The base of the pendulum operates in a permanent-magnet field, providing a one-turn torque generator. The null detector consists of a light source and a dual silicon photodiode. The p-layer of the silicon p-n junction is divided into two parts by a thin separation. When the base of the pendulum coincides with this separation, the null position is achieved, and the dc outputs of the dual photodiode are balanced. The servoamplifier used to control the pendulum to the null position is a standard commercial  $\mu A741$  integrated circuit. The lamp is rated for a useful life in excess of 20,000 hours. Severe environmental conditions have been applied both in test and in the field with no lamp or suspension failures.

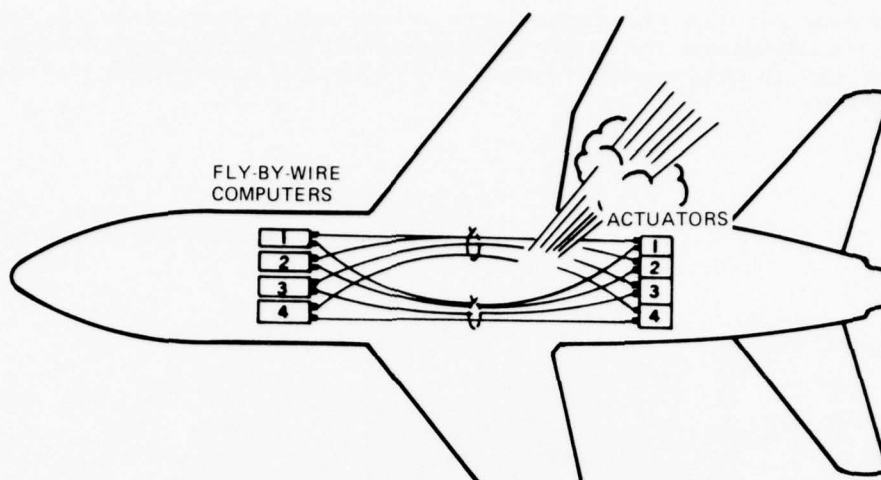
Studies have shown that spin motor and bearing failures account for about 65 percent of all rate gyro failures (Reference 16). Rate sensors used on future highly reliable automatic flight control systems will not use a spin motor and attendant bearings. Rate sensors that eliminate this weakness are currently being developed. These include a solid state rate sensor, a ring laser gyro and a magneto hydrodynamic rate sensor.

The General Electric "VYRO" solid state rate sensor eliminates the rotating mass with its associated bearings, motor and gimbal, and replaces them with a vibrating beam supported by two wires and driven by piezoelectric transducers. Without rotating parts, the VYRO is potentially more reliable than the conventional rate gyro. The predicted mean time between failure for the unit is 45,000 hours. Performance of the VYRO has been demonstrated on an F-4J airplane and in laboratory tests.

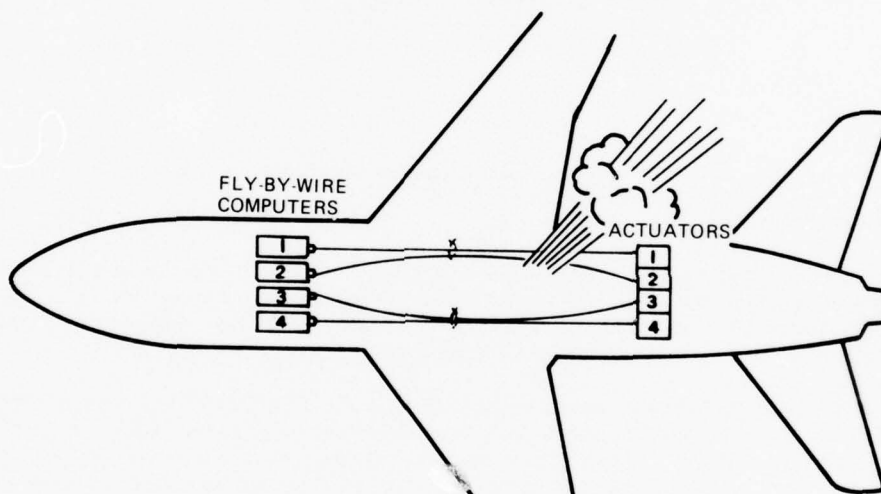
Because the ring laser rate sensor operation depends only on optical and electrical phenomena, it is mechanically simpler than conventional rotating-mass gyroscopes. A Honeywell ring laser gyro uses a laser block of dimensionally stable Cer-Vit material made by Owens-Illinois, Inc. (Reference 17). Electronic circuits in the gyro case control the laser and convert its optical output to an electrical signal. One of these gyros has operated continuously for more than 18,000 hours without any degradation in performance or increase in lasing threshold current. The Sperry Ring Laser Sensor is similar, but it uses an aluminum optical cavity that is heated to provide the required dimensional stability. The instrument case is evacuated and sealed, with all manufacturing adjustments locked prior to sealing.

The Honeywell magneto hydrodynamic rate sensor uses an angular accelerometer in the form of a torus of liquid metal as its basic sensor (Reference 15). The torus is continuously rotated about a diameter of the torus, which permits the device to measure angular rate in two axes. The complete rate sensor consists of a magneto hydrodynamic angular accelerometer, a

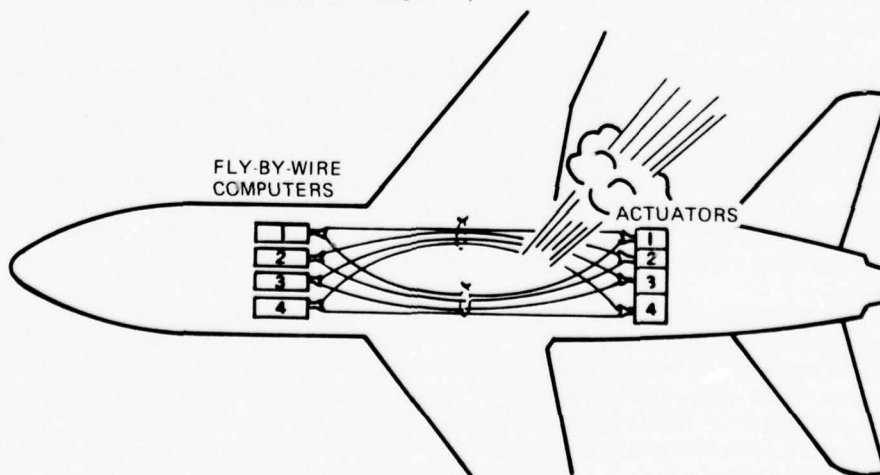




a. ATTEMPT AT DUAL WIRES FOR EACH CONNECTION (DAMAGE ON ONE SIDE OF AIRCRAFT)



b. CONVENTIONAL METHOD OF WIRING (DAMAGE ON ONE SIDE OF AIRCRAFT)



c. OPTICAL TRANSMISSION WITH SPLIT PATHS FOR EACH CHANNEL (INVULNERABILITY OF ALL FOUR CHANNELS TO DAMAGE ON ONE SIDE OF AIRCRAFT)

Figure 5. Quad-Redundant Fly-by-Wire System Wiring

synchronous hysteresis drive motor, a two-phase reference generator which permits resolution of the output into its two orthogonal axes, and a slip ring assembly to transfer the output signal from the rotating element to the preamplifier mounted within the hermetic seals. The sensor projects a very high reliability, but slip ring wearout requires a 1500-hour time between scheduled replacement.

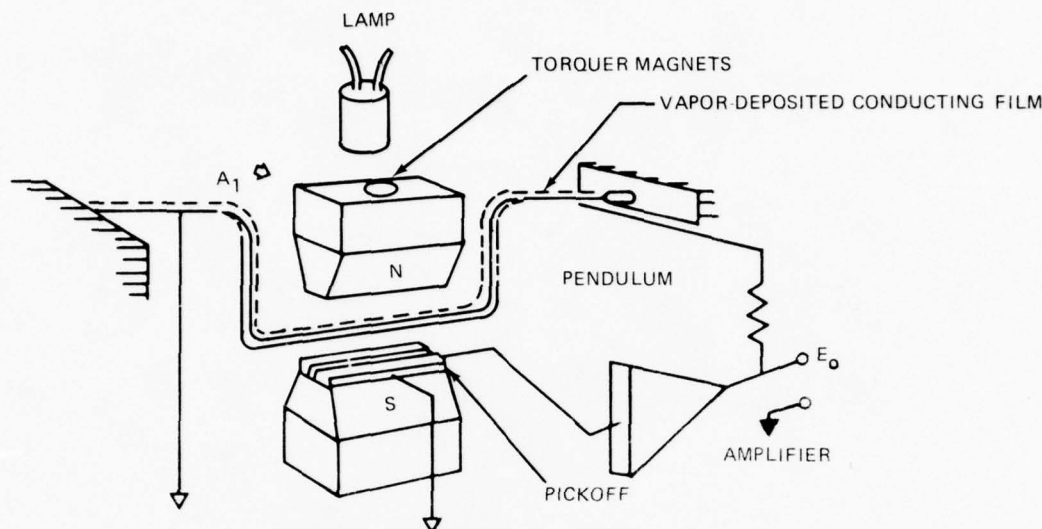


Figure 6. Honeywell Linear Accelerometer

### 3.4 Control Surface Actuators

Highly reliable automatic flight control systems will continue to use hydraulic actuators to drive the aerodynamic control surfaces. Trends in actuator technology are toward actuator redundancy schemes tailored to individual aircraft requirements, with the systems operating at pressures above the usual 3000 psi. Improvements will be made in hydraulic power supply units to increase reliability and reduce maintenance and downtime.

Perhaps the most significant changes in control surface actuation will come about in the hydraulic supply unit. Studies funded by the U.S. Navy show that significant improvements in system efficiency, maintainability, survivability and reliability can be attained by using 8000 psi supply pressures and incorporating advance filtration and distribution technology (Reference 18). The Navy lightweight hydraulic system concept is to use high pressure hydraulics to achieve large reductions in system weight and space requirements. This will be important in future high performance aircraft as aerodynamic considerations require high density airframes and thin wings, resulting in highly confined areas for installation of system components.

The hydraulic supply systems will incorporate advanced technologies, including those related to advanced hydraulic fluids, permanent tube connectors, hermetically sealed components, modularized components, metallic and double stage seals, titanium tubing and ultrafine system filtration. The Navy predicts a 20 to 25 percent savings in maintenance manhours and resulting aircraft downtime can be realized with the lightweight hydraulic system by the application of multiple stage seals and metallic designs which will extend replacement life in both static and dynamic conditions. The use of permanent type connectors will eliminate a large number of leakage paths and further reduce maintenance manhours required to service transmission lines and components. This will result in less introduction of contaminant into the system. Application of finer micron rated filters incorporating improved filter media will result in higher efficiency and contaminant holding capacity.

As aircraft rely more heavily on automatic flight control systems, the majority of commands to the control surface actuators will come from the automatic control channels. It is inevitable that the simpler, higher performance fly-by-wire implementation will result. One of the significant advantages of fly-by-wire is the potential elimination of primary control linkages, rods and cables. Future highly reliable control systems will employ integrated actuators, capable of positioning the control surface directly from an electrical command (Reference 15). The integrated actuator may contain some form of driver actuator, or it may be a straightforward electrohydraulic servomotor with integral monitoring elements. The integrated actuator could even contain its own electrically-powered hydraulic supply, perhaps using the controlled-displacement servopump principle.

The integrated actuators will probably use analog servo feedback loops, although research efforts now are directed toward development of digital valves and feedback encoders. Partial digital servoloops could be employed, but such hybrid systems are not competitive with analog at the present time.

#### 4.0 REDUNDANCY MANAGEMENT

Highly reliable flight control systems will continue to be achieved through redundancy management. The redundancy manager should think in system level terms at the onset to assure that the redundancy management philosophy considers all flight control system components. This should include secondary functions such as central air data systems and electrical and hydraulic power supplies. It is important to remember that complex redundancy management is only a means to achieve a reliability requirement on the system level. Future architectures will meet the high reliability requirements through both component reliability and redundancy management.

Past and present flight control systems have been treated as complete systems somewhat independent from the navigation system. Future systems will see an expansion of the systems engineering concepts to integrate the navigation and flight control systems. These two systems will be cross fed with each other through redundant digital data buses, such as the one defined by MIL-STD-1553. The Low Life Cycle Cost Avionics System (LLCCAS) program, currently being conducted by Boeing for the Air Force, is considering such a concept for the B-52G/H aircraft. Figure 7 illustrates a candidate LLCCAS architecture. The redundancy management architecture shown is a dual box configuration employing in-line monitoring.

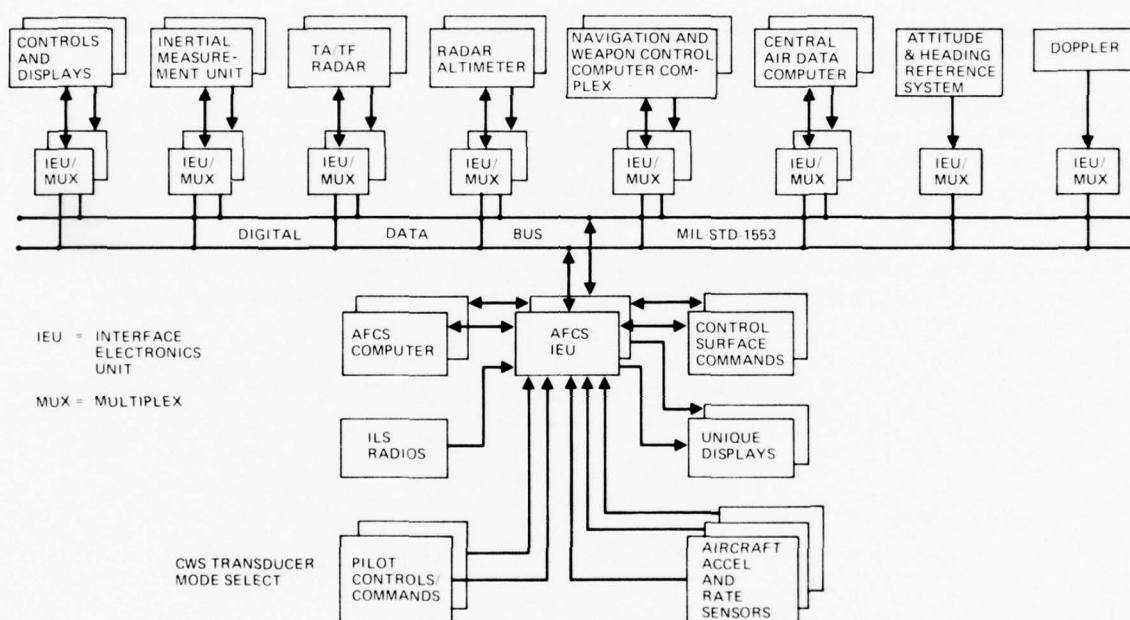


Figure 7. Low Life Cycle Cost Avionics Architecture

#### 4.1 Trends in Sensor Redundancy Management

Future sensor redundancy data management techniques will permit the reduction of sensor unit weight, power and volume by skewing the input axes of multiple sensors with respect to each other. The approach will be to apply these skewed sensor concepts to reduce the number of rate gyros required for a reliable and survivable fly-by-wire system (Reference 16). Rate gyros have been one of the higher failure rate items for contemporary systems.

In system concepts which include a general-purpose digital processor, the use of skewed sensor arrays may provide a significant redundancy management advantage (Reference 15). A digital processor is nearly mandatory because of the difficulty in accurately converting the skewed sensor data to the required orthogonal set for aircraft control in an analog computation implementation. Because of the great advantages of reducing the total sensor count in redundant systems, skewed sensor arrays should be included in the redundancy management trade studies.

Conventional flight control, attitude reference, and inertial systems have normally used orthogonal triads of gyros and accelerometers to obtain three-axis rate, attitude and/or velocity information. Redundant systems have been mechanized simply by duplicating the triads as necessary.

The skewed redundant strapped-down array is an efficient means for increasing reliability. The desired reliability level dictates the number of sensors which must be used in a system. The dual (or triple) redundancy of a five- (or six-) sensor array may be necessary to achieve the prescribed reliability level. Because the effective redundancy of dual or triple orthogonal sets may be achieved with pentad or hexad arrays which require fewer sensors, the overall system reliability is improved by the deletion of these relatively less reliable devices.

#### 4.2 Trends in Redundancy Management Within the Computer Mainframe

Redundancy management techniques use two general categories of failure detection, self-test and comparison test. The comparison tests, first used in redundant analog systems, will continue to be used in the future because of their proven track record.

The two basic approaches to comparison testing are (Reference 16):

- Cross-SSD (Signal Selection Device), which uses the output of an SSD as the good reference signal against which all other channels are compared.
- Cross-Channel, which compares each channel with the other channels and operates independently of any SSDs in the signal chain.

Typical cross-SSD and cross-channel monitors are shown in Figure 8, as they might appear in Channel 1 of a quadruplex configuration.

Future highly reliable flight control systems will improve system reliability by providing cross strapping of input signals. The SSD can, in principle, provide this function without monitoring the input signals; i.e., without removing a failed signal.

By detecting and removing a failed signal, failure detection can improve the benefits of an SSD relative to its cross strapping function. As an example, without failure detection the output of an SSD with four inputs will fail after two inputs have failed. With failure detection the output could conceivably fail only after four of the inputs have failed. There is an obvious trade off between the probability of two failures versus the probability of those combinations of detected and undetected failures and nuisance alarms which will cause disengagement of the device.

The second general category of failure detection is self-test and the related function of built-in test (BIT). Self-test architectures are currently in a constant state of revision and upgrading to improve efficiency and coverage.

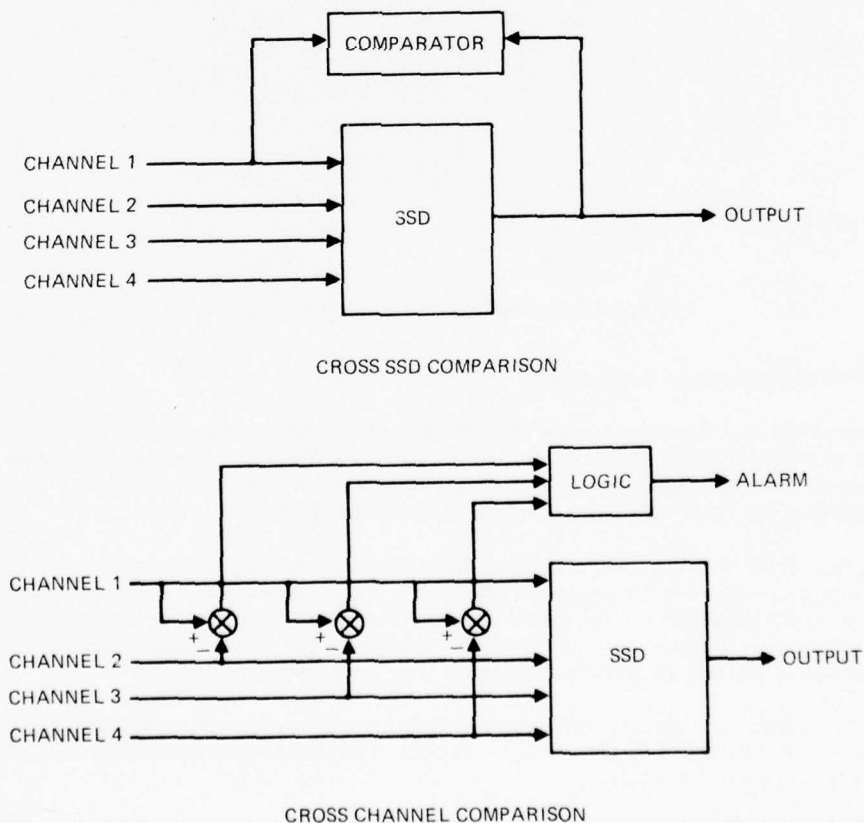


Figure 8. Comparison Monitoring Techniques



Self-testing of digital computers involves a mix of hardware and software. Certain basic portions of the computer must be operable before any self-testing can be conducted, e.g., power supplies and clocks. Failure of these basic portions must be detected by hardware.

With these basic portions of the computer operating, self-testing of the computer can begin. The design of the self-test program is based on the inverted pyramid test philosophy. That is, the program first tests the instructions that require a minimum of logic for their execution, and the memory locations that contain the self-test program. These verified instructions and memory locations are then used to test instructions and memory on the next higher level. This process is continued until all of the instructions, memory, and I/O have been verified.

The use of parity to monitor memory storage is becoming less popular among digital flight control system suppliers. It is doubtful that this use of parity will be required as part of redundancy management of future systems. The added expense and complexity for increased word length does not appear to be cost effective for future digital flight control systems.

Another area that impacts the digital redundancy management architecture is the question of synchronous or asynchronous operation. The trend appears to be moving toward asynchronous operation using equalization techniques to prevent numerical integration drift.

#### 4.3 Trends in Actuation Redundancy Management

This appears to be the least worked redundancy management area in terms of new architectures. The varied levels of actuation redundancy management have proven reasonably successful. Component reliability still remains as the weakest link and probably will continue to be in the near future.

Because of the fly-by-wire evolution, the state of the art is improving for redundant electrohydraulic servactuators. Four basic categories have been identified for the purposes of classifying redundant actuators (Reference 19). All categories apply to multiple channel, electrically commanded, closed loop, position control servos. Redundancy is assumed at the electrohydraulic interface but not necessarily for the power actuator. The four categories define techniques for combining the outputs of redundant servovalves and servactuators.

The first of the four categories identifies the general summing technique: averaging, active standby or voting. The second category defines the mechanical variable which is summed: position, velocity, or force. A third category classifies multistage servos by the type of power stage command: position commanded or velocity commanded. The last category classifies techniques for integrating dissimilar backup control channels: series or parallel.

High reliability requirements will continue to be met by the multicylinder hydraulic actuator. Two successful configurations are the tandem cylinder and the multiple single cylinders arranged side-by-side along the control surface hinge line. The tandem configuration has been built in dual and triple designs. Side-by-side configurations have been built as well as combinations of tandem and side-by-side applications to achieve "dual-dual" designs.

Further flexibility in overall actuation system concepts is afforded by the use of "split surfaces" — combinations of parallel, independent surfaces operated by individual (or multicylinder) actuators. The variations of actuator configurations and control surface arrangements are virtually limitless.

Actuation redundancy management is still the weakest reliability link in our future highly reliable flight controls systems. The next few years will see some improvements but not as dramatic as the computer architecture improvements.

#### 5.0 REFERENCES

1. Michael J. Popik, "Automated Landing System," Sperry Rand Engineering Review, 1971.
2. Seymour Salmirs and Harold N. Tobie, "Electronic Displays and Digital Automatic Control in Advanced Terminal Area Operations," AIAA Paper No. 74-27, January 30, 1974.
3. Clifford F. Newberry, "Design Freedom Offered by Fly-By-Wire," SAE 751044.
4. R. T. Jones and J. W. Nisbet, "Transonic Transport Wings - Oblique or Swept," Astronautics and Aeronautics, January, 1974.
5. Henry A. Shomber and Richard B. Holloway, "Advanced Controls For Commercial Transport Aircraft," SAE 740453.
6. B. M. Elson, "Hybrid Heavy-Lift Airships Under Study," Aviation Week and Space Technology, June 21, 1976.
7. Barry Miller, "Wider Use of Digital Techniques Seen," Aviation Week and Space Technology, November 10, 1975.

8. John McGough, Kurt Moses, Walter Platt, Gibson Reynolds and John Strole, "Digital Flight Control System Redundancy Study," AFFDL-TR-74-83, July, 1974.
9. E. E. Yore and D. C. Gunderson, "Active Control System Trends," NASA Symposium on Advanced Control Technology and Its Potential for Future Transport Aircraft, Los Angeles, California 1974.
10. David S. Hooker, Ira G. Pope, George R. Smith and George J. Vetsch, "Definition Study for an Advanced Fighter Digital Flight Control System," AFFDL-TR-75-59, June, 1975.
11. Lawrence Altman, "LSI Multiplies Design Options," Electronics, October 16, 1975.
12. Lucinda Mattera, "Monolithics Mature, Passives Improve," Electronics, October 16, 1975.
13. Jerry Lyman, "Film Carriers Win Productivity Prize," Electronics, October 16, 1975.
14. Stephen Osder and David LeFebre, "Final Report - Modification of Prototype Fly-By-Wire System to Investigate Fiber Optic Multiplexed Signal Transmission Techniques," AFFDL-TR-74-10, March, 1974.
15. Ronald Frazzini and Darrel Vaughn, "Analysis and Preliminary Design of an Advanced Technology Transport Flight Control System," NASA CR-2490, March, 1975.
16. C. R. Abrams and W. D. Weinstein, "The ASSET (Advanced Skewed Sensory Electronic Triad) Program," AGARD-CP-157-16, October 1974.
17. Philip J. Klass, "Laser Gyro Reemerges as INS Contender," Aviation Week and Space Technology, January 13, 1975.
18. Anon., "Program Plan - Lightweight Hydraulic System (LHS) for Aircraft," NADC, Air Vehicle Technology Dept., June, 1976.
19. Robert L. Anderson, "Synchronization of Multichannel Electrohydraulic Actuators - A Guide to the Design of Redundant Controls," Presented before the Seventy-Second Meeting of the Society of Automotive Engineers A-6 Committee, April 24-28, 1972.

**PART II**

**ANALYSIS AND TESTING**

## A SURVEY OF DESIGN METHODS FOR FAILURE DETECTION IN DYNAMIC SYSTEMS

Alan S. Willsky  
 Assistant Professor  
 Electronic Systems Laboratory  
 Department of Electrical Engineering and Computer Science  
 Massachusetts Institute of Technology  
 Cambridge, Massachusetts 02139

## Summary

In this paper we survey a number of methods for the detection of abrupt changes (such as failures) in stochastic dynamical systems. We concentrate on the class of linear systems, but the basic concepts, if not the detailed analyses, carry over to other classes of systems. The methods surveyed range from the design of specific failure-sensitive filters, to the use of statistical tests on filter innovations, to the development of jump process formulations. Tradeoffs in complexity versus performance are discussed.

## I. Introduction

With the increasing availability and decreasing cost of digital hardware and software, there has developed a desire in several disciplines for the development of sophisticated digital system design techniques that can greatly improve overall system performance. A good example of this can be found in the field of digital aircraft control (see, for example, Doolin [45], Taylor [46], and Meyer and Cicolani [47]), where a great deal of effort is being put into the design of aircraft with reduced static stability, flexible wings, etc. Such vehicles can provide improved performance in terms of drag reduction and decreased fuel consumption, but they also require sophisticated control systems to deal with problems such as active control of unstable aircraft, suppression of flutter, the detection of system failures, and management of system redundancy. The demands on such a control system are beyond the capabilities of conventional aircraft control system design techniques, and the use of digital techniques is essential.

Another example can be found in the field of electrocardiography. In recent years a great deal of effort has been devoted to the development of digital techniques for the automatic diagnosis of electrocardiograms (ECG's; see, for example, [47]). Such systems can be for preliminary screening of large numbers ECG's, for the monitoring of patients in a hospital, etc.

In this paper we review some of the recent work in one area of system theory that is of importance in both of these examples, as well as in many other system design problems. Specifically, we will discuss the problem of the detection of abrupt changes in dynamical systems. In the aircraft control problem one is concerned with the detection of actuator and sensor failures, while in the ECG analysis problem one wants to detect arrhythmias -- sudden changes in the rhythm of the heart. For the sake of simplicity in our discussion, we will refer to all such abrupt changes as "failures", although, as in the ECG example, the abrupt change need not be a physical failure. Our aim in this survey is to provide an overview of a number of the basic concepts in failure detection. The problem of system reorganization subsequent to the detection of a failure is considered in several of the references. We will point out these references in the sequel, but we will concentrate primarily on the detection problem.

The design of failure detection systems involves the consideration of several issues. One is usually interested in designing a system that will respond rapidly when a failure occurs; however, in high performance systems one often cannot tolerate significant degradation in performance during normal system operation. These two considerations are usually in conflict. That is, a system that is designed to respond quickly to certain abrupt changes must necessarily be sensitive to certain high frequency effects, and this in turn will tend to increase the sensitivity of the system to noise (via the occurrence of false alarms signaled by the failure detection system). The tradeoff between these design issues is best studied in the context of a specific example in which the costs of the various tradeoffs can be assessed. For example, one might be more willing to tolerate false alarms in a highly redundant system configuration than in a system without substantial back-up capabilities.

In general, one would like to design a failure detection system that takes system redundancy into account. For example, in a system containing several back-up subsystems we may be able to devise a simple detection algorithm that is easily implemented but yields only moderate false alarm rates. On the other hand, by implementing a more complex failure detection algorithm that takes careful account of system dynamics, one may be able to reduce requirements for costly hardware redundancy.

In addition to taking hardware issues into consideration, the designer of failure detection systems should consider the issue of computational complexity. One clearly needs a scheme that has reasonable storage and time requirements. It would also be useful to have a design methodology that admits a range of implementations, allowing a tradeoff study of system complexity vs. performance. In addition, it would be desirable to have a design that takes advantage of new computer capabilities and structures (e.g. designs that are amenable to modular or parallel implementations).

In this paper we survey a variety of failure detection methods, and, keeping the issues mentioned above in mind, we will comment on the characteristics, advantages, disadvantages, and tradeoffs involved in the various techniques. In order to provide this survey with some organization and to point out some of the key concepts in failure detection system design, we have defined several categories of failure detection systems and have placed the designs we have collected into these groups. Clearly such a grouping can only be a rough approximation, and we caution the reader against drawing too much of an inference about individual designs based on our classification of them (several of the techniques could easily fall into a number of our classes). In addition, for the sake of brevity we have limited our



detailed discussions to only a few of the many techniques. Our choice of those techniques has been motivated by a desire to span the range of available methods and by our familiarity with certain of these algorithms. Finally, we have attempted to collect all of those studies of the failure detection problem of which we are aware, and we apologize for any oversights.

## II. Formulations of the Failure Detection Problem

In this paper we are mostly concerned with the analysis of linear stochastic models in the standard state space form

### System Dynamic

$$x(k+1) = \Phi(k)x(k) + B(k)u(k) + w(k) \quad (1)$$

### Sensor Equation

$$z(k) = H(k)x(k) + J(k)u(k) + v(k) \quad (2)$$

where  $u$  is known input, and  $w$  and  $v$  are zero-mean, independent, white Gaussian sequences with covariances defined by

$$E[w(k)w'(j)] = Q\delta_{kj}, \quad E[v(k)v'(j)] = R\delta_{kj} \quad (3)$$

where  $\delta_{kj}$  is the Kronecker delta. We think of (1)-(3) as describing the "normal operation" or "no failure" model of the system of interest. If no failures occur, the optimal state estimator is given by the discrete Kalman filter equations [33]

$$\hat{x}(k+1|k) = \Phi(k)\hat{x}(k|k) + B(k)u(k) \quad (4)$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\gamma(k) \quad (5)$$

$$\gamma(k) = z(k) - H(k)\hat{x}(k|k-1) - J(k)u(k) \quad (6)$$

where  $\gamma$  is the zero-mean, Gaussian innovation process, and the gain  $K$  is calculated from the equations

$$P(k+1|k) = \Phi(k)P(k|k)\Phi'(k) + Q \quad (7)$$

$$V(k) = H(k)P(k|k-1)H'(k) + R \quad (8)$$

$$K(k) = P(k|k-1)H'(k)V^{-1}(k) \quad (9)$$

$$P(k|k) = P(k|k-1) - K(k)H(k)P(k|k-1) \quad (10)$$

Here  $P(i|j)$  is the estimation error covariance of the estimate  $\hat{x}(i|j)$ , and  $V(k)$  is the covariance of  $\gamma(k)$ . We refer to (4)-(10) as the "normal mode filter" in the sequel.

In addition to the above estimator, one may also have a closed loop control law, such as the linear law

$$u(k) = G(k)\hat{x}(k|k) \quad (11)$$

We then obtain the normal operation configuration depicted in Figure 1.

The problem of failure detection is concerned with the detection of abrupt changes in a system, as modeled by (1)-(3). Such abrupt changes can arise in a number of ways. For example, in aerospace applications, one is often concerned with the failure of control actuators and surfaces. Such abrupt changes can manifest themselves shifts in the control gain matrix  $B$ , increased process noise, or as a bias in equation (1) (as might arise if a thruster developed a leak [31]). In addition, failures of sensors may take the form of abrupt changes in  $H$ , increases in measurement noise, or as biases in (2). For simplicity, we will refer to abrupt changes in (1) as "actuator failures", and shifts in (2) will be called "sensor failures." Again we point out that in many applications shifts in (1) or (2) may be used to model changes in observed system behavior that have nothing to do with actuators or sensors.

The main task of a failure detection and compensation designs is to modify the normal mode configuration in order to include the capability of detecting abrupt changes and compensating for them by activating back-up systems, adjusting the feedback design appropriately, etc. Conceptually, we think of the detection-compensation system as part of the filtering portion of the feedback loop. As illustrated in Figures 2 and 3, the resulting filter design can take one of two forms. Either we perform a complete redesign of the filter, replacing (4)-(10) with a filter that is sensitive to failures, or we design a system that monitors the normal system configuration and adjusts the system accordingly. We will discuss examples of both of these structures.

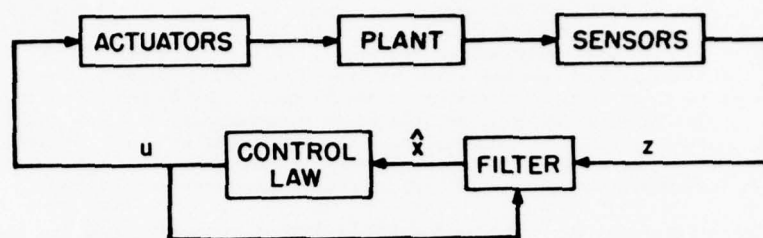


Figure 1: No-Failure System Configuration.

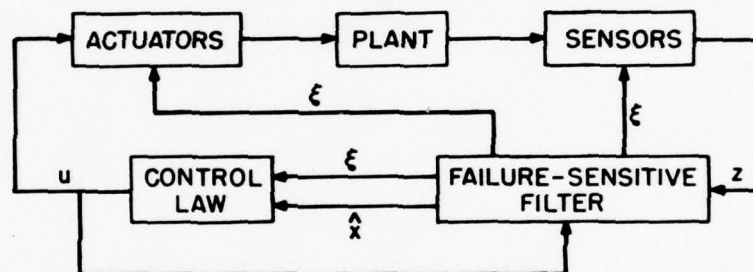


Figure 2: Failure Detection System Involving Failure-Sensitive Primary Filter (here  $\xi$  denotes information concerning detected failures).

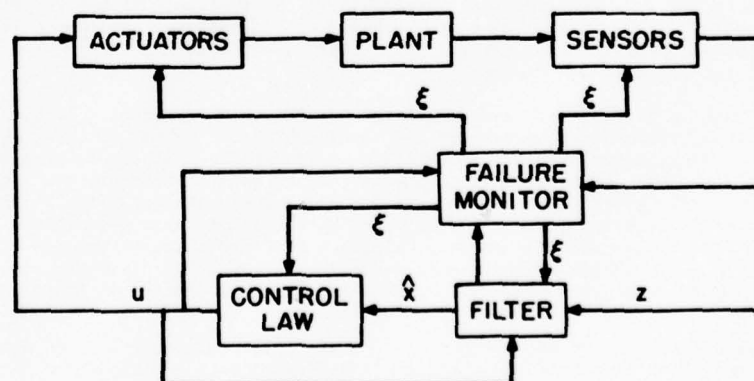


Figure 3: Failure Detection System Involving a Monitoring System for the No-Failure Configuration.

As mentioned earlier, we will concentrate primarily on the problem of failure detection, which we consider to consist of three tasks -- alarm, isolation, and estimation. The alarm task simply consists of making a binary decision -- either that something has gone wrong or that everything is fine. The problem of isolation is that of determining the source of the failure -- e.g., which sensor or actuator has failed, what type of arrhythmia has occurred, etc. Finally, the estimation problem involves the determination of the extent of failure. For example, a sensor may become completely non-operational (an "off" or "hard-over" failure), or it may simply suffer degradation in the form of a bias or increased inaccuracies, which may be modeled as an increase in the sensor noise covariance. In the latter case, estimates of the bias or the increase in noise may allow continued use of the sensor, albeit in a degraded mode. Clearly the extent to which we need to perform these various tasks depends upon the application. If a human operator is available, we may only be interested in generating an alarm that tells him to perform further tests. In other systems in which back-ups are available, we might settle for failure isolation without estimation. On the other hand, in the absence of hardware redundancy, we may be interested in using a degraded instrument and thus would need estimation information.

Intuitively we can associate increased software system complexity with the tasks -- i.e., isolation requires more sophisticated data processing than an alarm, and estimation more than isolation. On the other side, as we increase failure detection capabilities, we may be able to decrease hardware redundancy. Also, in some applications we may be able to delay isolation and estimation until after an alarm has been sounded. In such a sequential structure, one increases detector complexity after a failure has been detected, thereby reducing the computational burden during normal operation. Again the details of such considerations depend upon the particular application.

Another tradeoff involving failure detection system complexity involves its relation to detection system performance. For example, one might expect that one could achieve better alarm performance by using a priori knowledge concerning likely failure modes. That is, by looking for specific forms of system behavior that are characteristic of certain failures, one should be able to improve detection performance. Thus, it seems likely that alarm performance (as measured by the tradeoff between false alarms and missed detections) will be improved if we attempt simultaneous detection, isolation, and estimation of failures. This tradeoff of complexity vs. performance is extremely important in the design of failure detection systems.

In the following sections we will discuss several failure detection methods and will comment on their characteristics with respect to the issues mentioned in this and the preceding section. We have not provided a general set of failure models to be considered, as the various techniques are based on quite different failure models. These will be described as we discuss the various methodologies.

### III. "Failure-Sensitive" Filters

Our first class of failure detection concepts is aimed at overcoming the problem of an "oblivious filter". As has been noted by many authors [1]-[3], [33], the optimal filter defined by (4)-(10) performs well if there are no modelling errors; however, it is possible for the filter estimate to diverge if there are substantial unmodeled phenomena. The problem occurs because the filter "learns the state too well" -- i.e. the precomputed error covariance  $P$  and filter gain  $K$  become small, and the filter relies on old measurements for its estimates and is oblivious to new measurements. Thus, if an abrupt change occurs, the filter will respond quite sluggishly, yielding poor performance. Consequently, one would like to devise filter designs that remain sensitive to new data so that abrupt changes will be reflected in the filter behavior.

Two well-known techniques for keeping the filter sensitive to new data are the exponentially age-weighted filter studied Fagin [1] and Tarn and Zaborsky [2] and the limited memory filter proposed by Jazwinski [3]. Others, such as increasing noise covariances or simply fixing the filter gain are discussed by Jazwinski in [33]. These techniques yield only indirect failure information. That is, if an abrupt change occurs, these filters will respond faster than the normal filter, and one can base a failure detection decision on sudden changes of  $\hat{x}$ .

It is important to note a performance tradeoff evident in this method. As we increase our sensitivity to new data, (by effectively increasing the bandwidth of the Kalman filter), our system becomes more sensitive to sensor noise, and the performance of the filter under no-failure conditions degrades. In some cases this can be rather severe, and one may not be able to tolerate the degradation in overall system performance under no-failure conditions. One might then consider a two filter system -- the normal mode filter (4)-(10) as the primary filter, with this type of failure-sensitive filter as an auxiliary monitor, used only to detect abrupt changes. We remark that the tradeoff between detection performance and filter behavior under normal conditions is a characteristic of all failure detection systems and is analogous to the costs associated with false alarms and missed detection in standard detection problems [41].

The techniques mentioned so far in this section are rather indirect failure detection approaches. Several methods have been developed for the design of filters that are sensitive to specific failures. One method involves the inclusion of several "failure states" in the dynamic model (1)-(3). Kerr [25] has considered a procedure in which failure modes, such as the onset of biases, are included as state variables. If the estimates of these variables vary markedly from their nominal values, a failure is declared. A two-confidence interval overlap decision rule for failure detection using such failure state is described and its performance is analyzed in [25]. Note that this approach provides failure isolation and estimation at the expense of increased dimensionality and some performance degradation under no-failure conditions (inclusion of the added states effectively opens up the bandwidth of the Kalman filter).

An alternative to the addition of failure states to the dynamic model is the class of detector filters developed by Beard [4] and Jones [5]. Their work has led to a systematic design procedure for the detection of a wide variety of abrupt changes in linear time-invariant systems. They consider the continuous-time, time-invariant, deterministic system model

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) \quad (11)$$

$$z(t) = C\hat{x}(t) \quad (12)$$

and design a filter of the form

$$\frac{d}{dt} \hat{x}(t) = A\hat{x}(t) + D(z(t) - C\hat{x}(t)) + Bu(t) \quad (13)$$

The primary criterion in the choice of the gain matrix  $D$  is not that (13) provide a good estimate  $\hat{x}$  (as it is with observers or optimal estimators), but rather that the effects of certain failures are accentuated in the filter residual

$$\gamma(t) = z(t) - C\hat{x}(t) \quad (14)$$

The basic idea is to choose  $D$  so that particular failure modes manifest themselves as residuals which remain in a fixed direction or in a fixed plane.

To illustrate the Beard-Jones approach, let us consider a simple example from [4]. Suppose we wish to detect a failure of the  $i$ th actuator (i.e. in the actuator driven by the  $i$ th component of  $u$ ). If we assume the failure takes the form of a constant bias, our state equation becomes

$$\begin{aligned} \dot{\hat{x}}(t) &= A\hat{x}(t) + B[u(t) + v e_i] \\ &= A\hat{x}(t) + Bu(t) + v b_i, \quad t \geq t_0 \end{aligned} \quad (15)$$

where  $e_i$  is the  $i$ th standard basis vector,  $b_i$  is the  $i$ th column of  $B$ , and  $t_0$  is the (unknown) time of failure. Suppose we consider the case of full state measurement -- i.e., let  $C=I$ . In this case we obtain a differential equation for the residual

$$\dot{\gamma}(t) = [A-D]\gamma(t) + b_i \quad (16)$$

If we choose  $D=0I + A$ , we obtain

$$\begin{aligned} \dot{\gamma}(t) &= -\sigma\gamma(t) + v b_i \\ \gamma(t) &= e^{-\sigma(t-t_0)} \gamma(t_0) + \frac{v[1-e^{-\sigma t}]}{\sigma} b_i \end{aligned} \quad (17)$$

Thus, as the effect of the initial condition dies out,  $\gamma(t)$  maintains a fixed direction ( $b_i$ ) with magnitude proportional to failure size ( $v$ ). Note that as we increase  $\sigma$  (thus increasing filter gain), the initial condition dies out faster, but the magnitude of steady-state value of  $\gamma$  decreases. Thus, if there is any noise in the system, we cannot make  $\sigma$  arbitrarily large.

In their work Beard and Jones consider the design of such filters for an extremely wide variety of failure modes, including actuator and sensor shifts and shifts in  $A$  and  $B$ . The initial deterministic analysis for all of these cases was considered by Beard [4], while a systematic design procedure is given by Jones [5] for the design of the gain  $D$  to allow detection of several failure modes. Jones' approach is quite geometric in nature, and his formulation allows one to gain considerable insight into the detection problem. As pointed out in [5], the gain selection problem is quite similar to the output decoupling problem and requires the introduction of the important concept of "mutually detectable failure modes" in order to answer the question of whether or not one can simultaneously distinguish between several types of failures. Thus the question of failure isolation is of central importance in the design methodology derived in [5].

The results in [4],[5] represent perhaps the most thorough study of the basic concepts underlying failure detection. The tradeoff between detection and filter performance is discussed in depth in [5] and an attempt is made in [4] to introduce the concept of the level of redundancy in a dynamical system.

As mentioned in the example, the basic design procedure is deterministic. However, in this simple example we can see how one can take noise into account. If the system (11),(12) contains noise, we have seen that one may not wish to make the scalar  $\sigma$  as large as possible. In fact, one could choose  $\sigma$  so as to minimize the mean-square estimation error in the detector filter when there is no failure. In his thesis [5], Jones describes a procedure in which one first chooses the structure of  $D$  for failure detection purposes and then chooses the remaining free parameters in order to minimize the estimation error covariance. Although this yields a suboptimal filter design, it may work quite well, as it did in the problem reported in [5].

In summary, the Jones-Beard design methodology is extremely useful conceptually, can be used to detect a wide variety of failures, and provides detailed failure isolation information. It is suboptimal as an estimator, and if this presents a serious problem, one might wish to use the detector filter as an auxiliary monitoring system. This appears to be only a minor drawback, and the major limitation of the approach is its applicability only to time-invariant systems.



## IV. Voting Systems

Voting techniques are often useful in systems that possess a high degree of parallel hardware redundancy. Memoryless voting methods can work quite well for the detection of "hard" or large failures, and the papers of Gilmore and McKern [6], Pejza [7], and Ephgrave [8] discuss the successful application of voting techniques to the detection of hard gyro failures in inertial navigation systems.

In standard voting schemes, one has (at least) three identical instruments. Simple logic is then developed to detect failures and eliminate faulty instruments, for example, if one of the three redundant signals differs markedly from the other two, the differing signal is eliminated. Recently, Broen [9] has developed a class of voter-estimators that possesses advantages relative to standard voting techniques. Consider the dynamical system

$$\mathbf{x}(k+1) = \Phi \mathbf{x}(k) \quad (18)$$

with a triply redundant set of sensors

$$\begin{aligned} y_1(k) &= H_1 \mathbf{x}(k) + v_1(k) \\ y_2(k) &= H_2 \mathbf{x}(k) + v_2(k) \\ y_3(k) &= H_3 \mathbf{x}(k) + v_3(k) \end{aligned} \quad (19)$$

Broen develops a set of recursive filter equations for computing the estimate  $\hat{\mathbf{x}}(k)$  that minimizes

$$J_k = \sum_{i=0}^k \sum_{j=1}^3 w_{ji} \gamma_j'(i) R_j^{-1} \gamma_j(i) \quad (20)$$

where  $R_j$  is the covariance of the measurement noise  $v_j$ , and  $\gamma_j$  is the innovations sequence

$$\gamma_j(i) = y_j(i) - H_j \Phi^{i-k} \hat{\mathbf{x}}(k) \quad (21)$$

Here  $w_{ji}$  is a function of  $y_1(i)$ ,  $y_2(i)$ ,  $y_3(i)$  which is large if  $y_j(i)$  is close to the other two  $y_m$  and is small if  $y_j(i)$  deviates greatly from the other two. In this way, one obtains a "soft" voting

procedure in which faulty sensors are smoothly removed from consideration. This greatly alleviates the cost of false alarms, but the price is the on-line computation of the filter gain (which is a function of the  $w_{ji}$ ). Note that in equation (19), Broen appears to allow the  $y_i$  to be physically different sensors (different  $H_i$ 's), but the analysis of his paper makes it clear that he requires identical sensors -- i.e.  $H_1 = H_2 = H_3$ .

Voting schemes are in general relatively easy to implement and usually provide fast detection of hard failures, but they are only applicable in systems possessing a high level of parallel redundancy. They do not in general take advantage of redundant information provided by unlike sensors, and thus cannot detect failures in single or even doubly redundant sensors. In addition, voting techniques can have difficulties in detecting "soft" failures (such as a small bias shift).

## V. Multiple Hypothesis Filter-Detectors

A rather large class of adaptive estimation and failure detection schemes involves the use of a "bank" of linear filters based on different hypotheses concerning the underlying system behavior. In the work of Athans and Willner [10] and Lainiotis [11], several different sets of system matrices are hypothesized. Filters for each of the models are constructed, and the innovations from the various filters are used to compute the conditional probability that each system model is the correct one. In this manner, one can do simultaneous system identification and state estimation. In addition, an abrupt change in the probabilities can be used to detect changes in true system behavior. This technique has been investigated in the context of the adaptive control of the F-8C digital fly-by-wire aircraft by Athans, Dunn, Greene, et al., [35] and also has been applied to the problem of classifying rhythms and detecting rhythm shifts in electrocardiograms. Extremely good results in the latter case are reported by Gustafson, Willsky, and Wang in [36].

Techniques involving multiple hypotheses have also been used to design failure detection systems. Montgomery, Caqlayan, and Price, [12], [13] have used such a technique for digital flight control systems and have studied its robustness in the presence of nonlinearities via simulations. Recently a technique involving a bank of observers has been devised [34], and a successful application to a hydrofoil sensor failure problem is reported by Clark, Posth, and Walton in [34]. Also, Willsky, Deyst, and Crawford [15], [16] have applied the methodology devised by Buxbaum and Haddad in [14] to study failure detection for an inertial navigation problem. We will briefly describe this technique to illustrate some of the concepts underlying the bank of filters approach. We also refer the reader to Wernersson [42] for a technique that is similar to that discussed in [16].

Consider the system

$$x(k+1) = \Phi(k)x(k) + w(k) \quad (22)$$

$$z(k) = H(k)x(k) + v(k) \quad (23)$$

We are interested in detecting sudden shifts in certain of the components of  $x$  (e.g., bias states). We model these shifts by choosing the distribution of  $w$  appropriately. Let  $\{f_1, \dots, f_r\}$  be the set of hypothesized failure directions. We then assume that  $w$  has a high probability of being the usual process noise and a small probability of including a burst of noise in each of the failure directions. Thus the density for  $w(k)$  is

$$p_0 N(0, Q) + \sum_{i=1}^r p_i N(0, Q + \sigma_i f_i f_i') \quad (24)$$

$$\sum_{i=0}^r p_i = 1, \quad p_0 \gg p_i \quad i=1, \dots, r \quad (25)$$

Here  $N(m, P)$  is a normal density with mean  $m$  and covariance  $P$ .

If we hypothesize such a density at each point in time and if we assume that  $x(0)$  is normally distributed, we have the following expression for the conditional density of  $x(k)$  given  $z(1), \dots, z(k)$ :

$$p(x, k) = \sum_{i_0=0}^r \dots \sum_{i_{k-1}=0}^r p_i N(\eta_i, P_i) \quad (26)$$

Here  $\underline{i} = (i_0, \dots, i_{k-1})$  and the density has the following interpretation. Let  $\underline{j} = (j_0, \dots, j_{k-1})$  be a random  $k$ -tuple where  $j_s = i$  if there is a shift in the  $f_i$  direction at time  $s$  ( $i=0$  is used to denote no shift). Then

$$p_{\underline{i}} = \Pr(\underline{j} = \underline{i} | z(1), \dots, z(k)) \quad (27)$$

and  $\eta_{\underline{i}}$  and  $P_{\underline{i}}$  are the mean and covariance of the Kalman filter designed assuming  $\underline{j} = \underline{i}$  (i.e. assuming  $w(s)$  has covariance  $Q + \sigma_{i_s} f_{i_s} f_{i_s}'$ ). The  $P_{\underline{i}}$  can be computed in a sequential manner as a function of the various filter innovations. We refer the reader to [14]-[16] for the details of the calculations.

Note that the implementation of (26) requires an exponentially growing bank of filters (there are  $(r+1)^k$  terms in (26)). To avoid this problem a number of approximation techniques have been proposed [14]-[16]. The one used in [16] involves hypothesizing shifts only once every  $N$  steps. At the end of each  $N$  step period we "fuse" the  $(r+1)$  densities into a single density and begin the procedure again. In this way we implement only  $(r+1)$  filters at any time. We note that the techniques devised in [10]-[12] do not involve growing banks of filters (as the number of hypothesized models do not grow in time). However, it is possible for all of the filters in the bank to become oblivious, and thus shifts between the hypotheses may go undetected (see [16], [36] for examples). The technique of periodic fusing of the densities and initiation of new bank effectively avoids this problem (as would designing the original bank using age-weighted filtering techniques).

The technique described above was applied to the problem of detecting gyro and accelerometer bias shifts in a time-varying inertial calibration and alignment system. The results of these tests are extremely impressive. This is not surprising, as the multiple hypothesis method computes precisely the quantities of interest-- the probabilities of all types of failures under consideration. The cost associated with such a high level of performance is an extremely complex failure detection system. Note, however, that the parallel structure of the system allows one to consider highly efficient parallel processing computer implementations. In addition, the use of reduced-order filters for the various failure hypotheses may increase the practicality of such a scheme, or one might consider the use of simpler detection-only system to detect failures, with a switch to a multiple hypothesis procedure for failure isolation and estimation after a failure has been detected.

However, even if such a failure detection scheme cannot be implemented in a particular application, it provides a useful benchmark for comparison with simpler techniques. In addition, by studying the simulation of a multiple hypothesis method, one can gain useful insight into the dynamics of failure propagation and detection (see the discussion in [16]).

McGarty [23] has developed a method for rejecting bad measurements that bears some similarity to the approach just discussed. Each measurement has a binary random variable  $g(k)$  associated with it. If  $g(k)=1$  the measurement is "good", (i.e. the measurement contains the signal of interest), while  $g(k)=0$  denotes a bad data point (the measurement is pure noise). McGarty devises a maximum likelihood approach for estimating the values of the exponentially growing set of possibilities ( $g(i)=1$  or  $0$ ,  $i=1, \dots, k$ ). He also allows these variables to have a sequential correlation (i.e. knowing that the present measurement is good or bad says something about the next observations). A computationally feasible approximation method is devised and simulation results are described. We refer the reader to [23] for details.

Recently, Athans, Whiting, and Gruber [51] have also considered the problem of designing an estimator that can detect and remove bad or false measurements. Their approach is Bayesian in nature -- i.e. an estimate is generated of the a posteriori probability that a given measurement is false. The method of calculation of these pseudo-probabilities is quite similar to that used in the other multiple hypothesis methods (see [10]-[14]). The reader is referred to [51] for details of the analysis and for a discussion of some successful simulation results.

## VI. Jump Process Formulations

The problem of the detection of abrupt changes in dynamical systems suggests the use of jump process techniques in devising system design methodologies (see [39], [49]-[50] for general results on jump processes). One models potential failures as jumps, characterized by a priori distributions which reflect initial information concerning failure rates. The size of the possible failures are usually taken to be known. One could, however, model failure magnitude, as a random variable. This leads to a compound jump process formulation which greatly complicates the desired analysis. In any event, taking such a jump process formulation, one can devise failure-sensitive control laws and methods for computing the conditional probability of failure. Control problems of this type have received a great deal of attention in the literature. Swarder, and Robinson [17]-[20], [37] and Ratner and Luenberger [21] have considered the design of control laws which take into account the possibility of sudden shifts in system matrices. The results they have obtained are for the full-state feedback problem with no system randomness other than the jumping of the system matrices among a finite set of possible matrices.

Davis [22] has utilized nonlinear estimation techniques to solve a fault detection problem. His formulation is as follows: consider the scalar stochastic equations

$$dx(t) = a(t)x(t)dt + g(t)dv(t) \quad (28)$$

$$dy(t) = h(t)x(t)dt + dw(t) \quad (29)$$

where  $w$  and  $v$  are independent Brownian motion processes and

$$a(t) = a_0(t) [1 - \xi(t)] + a_1(t) \xi(t) \quad (30)$$

where

$$\xi(t) = \begin{cases} 0 & t < T \\ 1 & t \geq T \end{cases} \quad (31)$$

and  $T$  is a random variable. Here we interpret  $a_0$  as the unfailed dynamics, and  $a_1$  represents the failure mode. Davis derives the optimal, infinite-dimensional equations for the computation of the conditional mean of  $x$  and the conditional probability

$$\hat{\xi}(t|t) = \Pr[t \geq T | y(s), 0 \leq s \leq t] \quad (32)$$

An implementable approximation is described in [22], but evaluation of its performance has not as yet been made.

Note that Davis' method leads to an estimate of  $x$  that is suboptimal to under no-failure conditions. Chien [24] has devised a jump process formulation that avoids this difficulty for the problem of the detection of a jump or a ramp in a gyro bias. He considers the dynamical model.

$$\dot{x}(t) = \omega x(t) + w(t) \quad (33)$$

where  $w$  is a white noise process. Three hypotheses are conjectured for the form of the gyro output

Normal Mode  $H_0$ :

$$z(t) = x(t) + v(t) \quad \forall t \quad (34)$$

Bias Mode  $H_1$ :

$$z(t) = x(t) + m\xi(t) + v(t) \quad t > T \quad (35)$$

Ramp Mode  $H_2$ :

$$z(t) = x(t) + n(t-T)\xi(t) + v(t) \quad t > T \quad (36)$$

where  $n$  and  $m$  are unknown constants,  $v$  is white noise,  $T$  is the time of failure, and  $\xi(t)$  is as in (31).

Chien's approach is as follows: design a filter based on  $H_0$  (which will thus yield the optimal estimate for  $t < T$ , assuming no false alarms occur), and determine the steady-state effect of the degradations  $H_1$  and  $H_2$  on the filter residuals. If one then hypothesizes a failure rate  $q$  -- i.e.

$$P(T > t) = e^{-qt} \quad (37)$$

and if one further assumes a nominal size for the bias  $m$ , one can then compute an approximate stochastic



differential equation for  $\Pr(H_1 | z(s), s \leq t)$ , in which the input to this equation is the residual  $\gamma$  of the  $H_0$  filter. The details of the analysis are described in [24].

For his problem Chien is able to demonstrate that his detection procedure -- based on the assumption of a nominal value for the bias failure  $m$  -- has the capability of detecting biases larger than  $m$  and also can be used to detect ramps (mode  $H_2$ ). Of course, the delay times until detection in these cases are greater than if one implemented a filter based on the proper bias size or if one were looking for a ramp (indicating the potential usefulness of estimating the failure magnitude). The major advantages of Chien's approach are the simplicity of the detector (implementation of a scalar stochastic equation) and the fact that one obtains an estimate of precisely the quantity of interest -- the conditional probability of failure. The simplicity of the scheme may, in fact, make it a great deal more robust in the face of system modelling errors (such as the use of an extremely simplified gyro error model) than more sophisticated approaches. Also, this approach leads to no degradation in performance prior to detection of the failure. In addition, the use of a probabilistic description of the time of failure allows one to avoid the problem of the oblivious filter -- i.e. the fact that a failure can occur at any time has been incorporated in the design, which therefore will remain sensitive to new data.

The drawbacks of the scheme are the use of a fixed bias size and the use of the steady-state effect of the failure on the filter residual. The first of these may not be too much of a problem (as Chien has pointed out), but the second may cause difficulties. Specifically, this limits the approach to time-invariant systems and filters. In addition, as the transient effect of the failure has been ignored, it may be difficult to make quick detections of certain changes (i.e. we may have to wait until the transient dies out). In the next section we will discuss an approach (the GLR method) which has several concepts in common with Chien's approach and which allows one to overcome these two drawbacks (at the cost of added computational complexity, of course).

In summary, jump process formulations appear to be quite natural for failure detection problems. One usually makes approximations in the analysis in order to obtain implementable solutions. These simplifications impose some limitations on the capabilities of the designs, but there is at present no systematic analytical procedure for evaluating these limitations or for studying tradeoffs between design complexity and system performance.

#### VII. Innovations-Based Detection Systems

Chien's failure detection technique can also be placed in the class of failure detection methods that involve the monitoring of the innovations of a filter based on the hypothesis of normal system operation. In such a configuration the overall system uses the normal filter until the innovations monitoring system detects some form of aberrant behavior. The fact that the monitoring system can be attached to a filter-controller feedback system is particularly appealing, since overall system behavior is not disturbed until after the monitor signals a failure and since the monitoring system can be designed to be added to an existing system.

Mehra and Peschon [26] have suggested a number of possible statistical tests to be performed on the innovations. One of these is a chi-squared test which was applied in [15], [16] by Willsky, Deyst and Crawford. Let  $\gamma(k)$  be the  $p$ -dimensional innovations for the filter defined by (4)-(10). If the system is operating normally, the innovations is zero-mean and white with known covariance  $V(k)$ . In this case the quantity

$$\ell(k) = \sum_{j=k-N+1}^k \gamma'(j) V^{-1}(j) \gamma(j) \quad (38)$$

is a chi-squared random variable with  $Np$  degrees of freedom [26], [15], [16]. If a system abnormality occurs, the statistics of  $\gamma$  change, and one can consider a detection rule of the form

$$\ell(k) \begin{matrix} > \\ \geq \end{matrix} \begin{matrix} \text{FAILURE} \\ \text{NO FAILURE} \end{matrix} \quad \epsilon \quad (39)$$

With the aid of chi-squared tables, one can compute the probability  $P_F$  of false alarm as a function of the innovations window length  $N$  and the decision threshold  $\epsilon$ . The probability  $P_D$  of correct detection depends upon the particular failure mode (see [16] and the discussion of the GLR approach to follow). We note that for a given failure mode, as  $N$  increases the detection probability may decrease -- i.e. by averaging a larger number of residuals we smooth out the effect of a failure on  $\gamma$ , and the detector may become somewhat oblivious (or at the very best responds quite slowly) to new data. On the other hand, too small a value of  $N$  may yield an unacceptably high value of  $P_F$ .

The implementation of the chi-squared test (38), (39) is quite simple, but, as one might expect, one pays for this simplicity with rather severe limitations on performance. As described in [15], [16] this method was applied to the same inertial calibration and alignment problem to which the Buxbaum-Haddad multiple hypothesis approach [14]-[16], described in Section V was applied. The performance of the chi-squared test was mixed. The method is basically an alarm method -- i.e. the system (38), (39) makes no attempt to isolate failures -- and one finds that those failure modes that have dramatic effects on  $\gamma$  are detectable by this method; however more subtle failures are more difficult to detect with simple scheme. Comparing the performance of the multiple hypothesis and chi-squared systems, we see that in some cases we can obtain superior alarm capabilities if we simultaneously attempt to do failure isolation and estimation. One can obtain some failure isolation information by considering the components of  $\gamma$  separately (this may be especially useful for sensor failures), and we refer the reader to [15], [16] for a detailed discussion of this and other aspects of the chi-squared method.



Another innovations-based approach, developed by Merrill [27], is motivated by a desire to suppress bad sensor data. Merrill devises a modification of the least squares criterion in order to suppress extremely large residuals (which are given a very large weighting in the usual least squares framework), and he applies his methodology to a power system application.

A final technique in this category has been studied by several researchers -- Willsky and Jones [28],[29], McAulay and Denlinger [30], Deyst and Deckert [31], Sanyaland Shen [32], and Chow, Dunn and Willsky [38]-- and we will describe the most general formulation of the approach, developed in [28],[29]. This technique, which we call the generalized likelihood ratio (GLR) approach, was in part motivated by the shortcomings of the simpler chi-squared procedure. The GLR approach, which can be applied to a wide range of actuator and sensor failures, makes an attempt to isolate different failures by using knowledge of the different effects such failures have on the system innovations. The method provides an optimum decision rule for failure detection and provides useful failure identification information for use in system reorganization subsequent to the detection of a failure. In addition, one can devise a number of simplifications of the technique and can study analytically the tradeoff between GLR complexity and GLR performance.

Consider the basic dynamical model (1)-(3). The following are 4 possible modifications of these equations that incorporate certain sudden system changes (see Willsky and Jones [28],[29] and Gustafson, Willsky, and Wang [36] for physical motivation for these and other failure modes of the same general type):

#### Dynamic Jump

$$x(k+1) = \Phi(k)x(k) + B(k)u(k) + w(k) + v\delta_{k+1,\theta} \quad (40)$$

Here  $v$  is an unknown  $n$ -vector,  $\theta$  is the unknown time of failure, and  $\delta_{ij}$  is the Kronecker delta. Such a model can be used to model sudden shifts in bias states (as in the inertial problem studied in [15], [16]).

#### Dynamic Step

$$x(k+1) = \Phi(k)x(k) + B(k)u(k) + w(k) + v\sigma_{k+1,\theta} \quad (41)$$

Here  $\sigma_{ij}$  is the unit step

$$\sigma_{ij} = \begin{cases} 1 & i \geq j \\ 0 & i < j \end{cases} \quad (42)$$

This model can be used to model certain actuator failures (compare to the Beard-Jones example in Section III; see equation (15)).

#### Sensor Jump

$$z(k) = Hx(k) + Ju(k) + v(k) + v\delta_{k,\theta} \quad (43)$$

We can use this to model bad data points.

#### Sensor step

$$z(k) = Hx(k) + Ju(k) + v(k) + v\sigma_{k,\theta} \quad (44)$$

Sudden changes in sensor biases fit into this model.

By the linearity of the system (1)-(3) and the filter (4)-(10), one can determine the effect of each of the failure modes on the innovations. The general form is

$$\gamma(k) = G(k;\theta)v + \tilde{\gamma}(k) \quad (45)$$

where  $\tilde{\gamma}(k)$  is the filter innovations if no failure occurs, and the matrix  $G$  can be precomputed (see [29], [38]). This matrix, which is different for each of the four cases (40)-(44), is called the failure signature matrix and provides us with an explicit description of how various failures propagate through the system and filter.

The full-blown GLR method involves the following: we assume we are looking for one of the four classes of failures and have computed the appropriate signature matrix. Given the residuals, we compute the maximum likelihood estimates of  $v$  and  $\theta$ , and, assuming that these estimates are correct, we compute the log-likelihood ratio for failure versus no failure (see Van Trees [41] for a general discussion of GLR methods). The implementation of the full GLR requires a linearly growing bank of matched filters, computing the best estimates of  $v$  assuming a particular value of  $\theta \in \{1, \dots, k\}$ .

A number of remarks can be made concerning the GLR system. We note that, as with other methods such as Buxbaum-Haddad or Chien, the inclusion of the variable  $\theta$  to indicate our uncertainty as to the time of failure keeps the detection system sensitive to new data. However, it is the estimation of  $\theta$  that causes the growing complexity problem. On the other hand, even if the full GLR is not implementable, it can serve as a benchmark for other schemes and can in fact be used as a starting point for the design of simpler systems. One simplification that eliminates the growing complexity is the restriction of the estimate of  $\theta$  to a window

$$k-N \leq \theta \leq k-M \quad (45)$$

where the lower bound is included to limit complexity, and the upper bound is set by failure observability and false alarm considerations. Successful simulation runs with  $N=M$  (i.e., when we don't optimize  $\theta$  at all and have only one matched filter for  $v$ ) are reported by Willsky and Jones in [29]. We remark only that the price one pays for "windowing" the estimate of  $\theta$  is in a reduction in the accuracy of the estimate of  $v$ . For example, in the case of  $N=M$ , we often are able to detect failures extremely quickly, but if  $\hat{\theta}=k-N$  is not the correct time of failure, the estimate of  $v$  may be severely degraded (e.g., our estimate of the slope of a ramp changes as we change our estimate of the time at which it started). We note that the estimation of  $\theta$  is similar to time-of-arrival estimation problems that arise in various applications, and refer the reader to Van Trees [44] for a general discussion of several techniques.

Also, we note that even if the physical system and filter are time-invariant, the GLR monitoring system is time-varying, as the failure signature  $G$  includes transient effects. In some cases one may be able to neglect these and utilize a simpler steady state signature. This is quite similar to Chien's use [24] of the steady-state effect of the failure on the residuals, and the criticisms of that approach, given in Section VI, apply here as well.

One can also simplify the implementation by either partially or completely specifying the failure magnitude  $v$ . Constrained GLR (CGLR) is based on the assumption that

$$v = \alpha f_i \quad (46)$$

where  $\alpha$  is an unknown scalar and  $f_i$  is one of  $r$  possible failure directions. This technique is described in [29]. If we completely specify  $v$

$$v = v_0 \quad (47)$$

we obtain the simplified GLR (SGLR) algorithm which is extremely simple to implement, as we have completely eliminated the need for the matched filters to estimate  $v$ . The use of specified failure sizes is similar to that proposed by Chien [24], although in SGLR one can use the time-varying failure signature, which should aid in failure detection. As initial results for the detection of electrocardiogram arrhythmias, indicate (see Gustafson, et.al., [36]) the estimation of  $v$  is not nearly as important for detection as the matching of failure signatures. Also, by the use of several values of  $v_0$  (i.e. by implementing several parallel SGLR's), one can achieve a high level of failure isolation without a great deal of additional software complexity. In addition, one could consider a "dual-mode" procedure in which SGLR is used for alarm and isolation, with full GLR used only afterward in order to estimate the magnitude of the failure.

The various simplifications of GLR, as well as full GLR, are amenable to certain analysis, such as the calculation of  $P_F$ ,  $P_D$  and (at least for SGLR) the expected time delay in detection. By performing such analyses, one can study in detail the tradeoff between complexity and performance. A methodology for such comparisons is presently being developed and is being applied to an aircraft failure detection problem. Initial results are reported by Chow, et.al., in [38], and a description of a detailed methodology will be reported in the near future. (see Bueno, Chow, Gershwin, and Willsky [43]). In addition, to the calculation of  $P_F$  and  $P_D$ , the comparison methodology reported in [43] includes the computation of cross-detection probabilities -- i.e. the probability of detecting a failure of type A when a failure of type B has occurred. Such information can be useful in designing failure isolation procedures and also in determining if failure detector A can be successfully utilized as an alarm for failures of type B. This can lead to substantial simplifications in a failure alarm system. Also, we refer the reader to [29], [36], and [38] for successful simulations of the GLR method.

Presently the GLR method is being extended to other failure modes, such as:

#### Hard-Over Actuator Failure

$$x(k+1) = \Phi(k)x(k) + [B + M\sigma_{k+1,\theta}]u(k) + w(k) \quad (48)$$

With this model we can take into account complete (or "off") failures of certain actuators. For example an off failure of the  $i$ th actuator can be modeled by choosing  $M$  all zero except for the  $i$ th column, which is taken to be the negative of the  $i$ th column of  $B$ . The GLR detector for (48) is presently under development [38], [43], and we note that this model is more difficult than the others as the effect of the failure is modulated by the input values  $u(k)$ .

#### Increased Process Noise Failures

$$x(k+1) = \Phi(k)x(k) + B(k)u(k) + w(k) + \xi(k)\sigma_{k+1,\theta} \quad (49)$$

Here  $\xi$  is additional white process noise.

#### Hard-Over Sensor Failures

$$x(k+1) = Hx(k) + Ju(k) + v(k) + [Mx(k) + Su(k)]\sigma_{k,\theta} \quad (50)$$

Here the failures are modulated by  $u$  and  $x$ , and a failure of the  $i$ th sensor is modeled by choosing the

ith rows of M and S appropriately.

#### Added Sensor Noise Failure

$$z(k) = Hx(k) + Ju(k) + v(k) + \xi(k)\sigma_{k+1,0} \quad (51)$$

The analysis of these failure modes is presently being performed [38],[43], and it is anticipated that SGLR algorithms will also be developed.

In addition to these failure modes, one can develop additional models along these lines for particular applications. In particular, we have developed several additional models similar to those described by equations (40)-(44) for our work on the detection and classification of arrhythmias in electrocardiograms. The results reported in [36] are rather striking, as in all the tests performed we observed no false alarms, detected all rhythm changes immediately (with no incorrect estimates of  $\theta$ ), and classified all rhythm changes correctly. These tests utilized the full GLR approach and have provided useful insight into the characteristics of the method. For example, the use of maximum likelihood estimates of  $v$  and  $\theta$  precludes the use of a priori statistics on these variables. In the ECG problem, one is quite interested in accurate estimates of  $v$ , and one also can come up with reasonable a priori statistics on  $v$  based on physical arguments. Thus, it may pay to incorporate such a priori statistics into the GLR system, and this can be done rather easily by proper initialization of the matched filters estimating  $v$ . On the other hand, for the ECG problem one does not want to look for abrupt changes at one point in the record more than at another, and thus it does not make sense to include a priori statistics on  $\theta$ . In fact, one can argue that inclusion of a priori failure information tends to discount the observed data in order to avoid false alarms (unless failures are extremely likely), and one should probably avoid the inclusion of such information unless one is especially worried about false alarms. However, if one wishes to use such data, one can utilize the interpretation of the likelihood ratios as ratios of conditional probabilities of failure time in order to determine the appropriate modification of GLR [29].

Finally, we note the the GLR system provides extremely useful information for system compensation subsequent to the detection of a failure. For example, one can utilize the GLR-produced estimates of  $v$  and  $\theta$  to determine an optimal update procedure for the filter estimate and covariance [29]. Once this update has been performed, the GLR system can be used to detect further failures, thus allowing the detection of multiple events. We refer the reader to [29],[38] for further discussions of the use of GLR-produced information in the design of failure compensation systems.

#### VIII. Conclusions

In this paper we have discussed a number of the issues involved in the design of failure detection systems. We have also reviewed a variety of existing failure detection methods and have discussed their characteristics and design tradeoffs. The failure detection problem is an extremely complex one, and the choice of an appropriate design depends heavily on the particular application. Issues such as available computational facilities and level of hardware redundancy enter in a crucial way in the design decision. For example, as we have mentioned, the use of a sophisticated failure detection-compensation system may allow one to reduce the level of hardware redundancy without much of a loss in overall system reliability.

The development of failure detection method is still a relatively new subject. At this time most of the work has been at a theoretical level with only a few real applications of techniques [6]-[9], [13], [31], [36]. Much work is yet to be done in the development of implementable systems complete with a variety of design tradeoffs. Work is needed in the development of efficient techniques for failure compensation and system reorganization. In addition, there is a great need for the analysis of the robustness of various failure detection systems in the presence of variations in system parameters and in the presence of modeling errors and system nonlinearities. For example, it is conjectured that SGLR is less sensitive to parameter errors than the more complex full GLR; however, at present there are no analytical results or simulations to support this conjecture. These and other issues, such as the incorporation of fault-tolerant computer concepts into an overall reliable design methodology (see Deyst [40]) await investigation in the future.

#### References

1. Fagin, S.L., "Recursive Linear Regression Theory, Optimal Filter Theory, and Error Analyses of Optimal Systems," IEEE International Convention Record, March 1964, pp. 216-240.
2. Tarn, T.J. and Zaborszky, J., "A Practical Nondiverging Filter," AIAA Journal, Vol. 8, No.6, June 1970, pp. 1127-1133.
3. Jazwinski, A.H., "Limited Memory Optimal Filtering", IEEE Trans. on Aut. Cont., Vol. 13, 1968, pp. 558-563.
4. Beard, R.V., Failure Accommodation in Linear Systems Through Self-Reorganization, Rept. MVT-71-1, Man Vehicle Laboratory, Cambridge, Massachusetts, February 1971.
5. Jones, H.L., Failure Detection in Linear Systems, Ph.D. Thesis, Dept. of Aeronautics and Astronautics, M.I.T., Cambridge, Mass., Sept. 1973.
6. Gilmore, J. and McKern, R., "A Redundant Strapdown Inertial System Mechanization -- SIRU," presented at the AIAA Guidance, Control and Flight Mechanics Conferences, Santa Barbara, Calif., Aug., 17-19, 1970.
7. Pejisa, A.J., Optimum Orientation and Accuracy of Redundant Sensor Arrays, Honeywell Aerospace Div.,



Minneapolis, Minn, 1971.

8. Ephgrave, J.T., Redundant Adaptive Strapdown Navigation Systems, Aerospace Report No. TOR-0066(5306)-10, The Aerospace Corp., Oct. 31, 1969.
9. Broen, R.B., "A Nonlinear Voter-Estimator for Redundant Systems," Proc. of the 1974 IEEE Conference on Decision and Control, Phoenix, Arizona, pp. 743-748.
10. Athans, M. and Willner, D., "A Practical Scheme for Adaptive Aircraft Flight Control Systems", Symp. on Parameter Estimation Techniques and Applications in Aircraft Flight Testing, NASA Flt. Res. Ctr., Edwards AFB, April 24, 25, 1973.
11. Lainiotis, D.G., "Joint Detection, Estimation, and System Identification", Information and Control, Vol. 19, No.1, Aug. 1971, pp.75-92.
12. Montgomery, R.C., and Caglayan, A.K., "A Self-Reorganizing Digital Flight Control System for Aircraft," AIAA 12th Aerospace Sciences Meeting, Washington, D.C., Jan. 30 - Feb. 1, 1974.
13. Montgomery, R.C. and Price, D.B., "Management of Analytical Redundancy in Digital Flight Control Systems for Aircraft," AIAA Mechanics and Control of Flight Conference, Anaheim California, August 5-9, 1974.
14. Buxbaum, P.J. and Haddad, R.A., "Recursive Optimal Estimation for a Class of Nongaussian Processes", Proc. of Symp. on Computer Processing in Communications, Polytech Inst. of Brooklyn, April 8-10, 1969.
15. Willsky, A.S., Deyst, J.J., and Crawford, B.S., "Adaptive Filtering and Self-Test Methods for Failure Detection and Compensation", Proc. of the 1974 JACC, Austin, Texas, June 19-21, 1974.
16. Willsky, A.S., Deyst, J.J., and Crawford, B.S., "Two Self-Test Methods Applied to an Inertial System Problem," J. Spacecraft and Rockets, Vol. 12, No.7, July 1975, pp. 434-437.
17. Pierce, B.D. and Swarder, D.D., "Bayes and Minimax Controllers for a Linear System with Stochastic Jump Parameters," IEEE Trans. on Automatic Control, Vol. AC-16, No.4, Aug. 1971, pp. 300-307.
18. Swarder, D.D., "Bayes' Controllers with Memory for a Linear System with Jump Parameters", IEEE Trans. on Automatic Control, Vol. AC-17, Feb. 1972, pp. 118-121.
19. Swarder, D.D. and Robinson, V.G., "Feedback Regulators for Jump Parameter Systems with State and Control Dependent Transition Rates," IEEE Trans. on Automatic Control, Vol. AC-18, No. 4, Aug. 1973, pp. 355-360.
20. Robinson, V.G. and Swarder, D.D., "A Computational Algorithm for Design of Regulators for Linear Jump Parameter Systems," IEEE Trans. on Automatic Control, Vol. AC-19, Feb. 1974, pp. 47-49.
21. Ratner, R.S. and Luenberger, D.G., "Performance-Adaptive Renewal Policies for Linear Systems", IEEE Trans. on Automatic Control, Vol. AC-14, Aug. 1969, pp. 344-351.
22. Davis, M.H.A., "The Application of Nonlinear Filtering to Fault Detection in Linear Systems", IEEE Trans. on Automatic Control, Vol. AC-20, No.2, April 1975, pp. 257-259.
23. McGarty, T.P., "State Estimation with Faulty Measurements: An Application of Bayesian Outlier Rejection," Proc. of the Fifth Symposium on Nonlinear Estimation and Its Applications, San Diego, Calif., Sept. 1974.
24. Chien, T.T., An Adaptive Technique for a Redundant - Sensor Navigation System, Rept. T-560, Draper Labs., Cambridge, Mass., February 1972.
25. T.H. Kerr, "A Two Ellipsoid Overlap Test for Real Time Failure Detection and Isolation by Confidence Regions", Pittsburgh Conf. on Modelling and Simulation, April 24-26, 1974.
26. Mehra, R.K. and Peschon, J., "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems", Automatica, Vol. 7, pp. 637-640.
27. Merrill, H.M., Bad Data Suppression in State Estimation, with Applications to Problems in Power, Ph.D. Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., June 1972.
28. Willsky, A.S. and Jones, H.L., "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes," Proc. of the 1974 IEEE Conference on Decision and Control, Phoenix, Arizona, Nov. 1974.
29. Willsky, A.S. and Jones, H.L., "A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems", IEEE Trans. on Automatic Control, Vol. AC-21, Feb. 1976, pp. 108-112.
30. McAulay, R.J. and Denlinger, E., "A Decision-Directed Adaptive Tracker", IEEE Trans. on Aero. and Elec. Sys., Vol. AES-9, March 1973, pp. 229-236.
31. Deyst, J.J. and Deckert, J.C., "RCS Jet Failure Identification for the Space Shuttle", Proc. of IFAC 75, Cambridge, Mass., August 1975.
32. Sanyal, P. and Shen, C.N., "Bayes' Decision Rule for Rapid Detection and Adaptive Estimation Scheme with Space Applications", IEEE Trans. on Automatic Control, Vol. AC-19, June 1974, pp. 228-231.



33. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.
34. Clark, R.N., Fosth, D.C., and Walton, V.M., "Detecting Instrument Malfunctions in Control Systems", IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-11, No.4, July 1975, pp. 465-473. .
35. Athans, M., Dunn, K.-P., Greene, C.S., Lee, W.H., Sandell, N.R., Jr., Segall, I., and Willsky, A.S., "The Stochastic Control of the F-8C Aircraft Using the Multiple Model Adaptive Control (MMAC) Method", Proc. 1975 IEEE Conf. on Decision and Control, Houston, Texas, December 1975.
36. Gustafson, D.E., A.S. Willsky, and J.-Y. Wang, "Final Report: Cardiac Arrhythmia Detection and Classification Through Signal Analysis", The Charles Stark Draper Laboratory, Cambridge, Mass., Rept. No. R-920, July 1975.
37. Szwed, D.D., "Feedback Control of a Class of Linear Systems with Jump Parameters", IEEE Trans. on Automatic Control, Vol. AC-14, No.1, Feb., 1969, pp. 9-14.
38. Chow, E., Dunn, K.-P., and Willsky, A.S., "Research Status Report to NASA Langley Research Center: A Dual-Mode Generalized Likelihood Ratio Approach to Self-Reorganizing Digital Flight Control System Design", M.I.T. Electronic Systems Laboratory, Cambridge, Mass., April 1975.
39. Boel, R., Varaiya, R., and Wong, E., "Martingales on Jump Processes I: Representation Results", and "II: Applications", SIAM Journal on Control, Vol. 13, August 1975, pp. 999-1061.
40. Deyst, J.J., "A Design Approach to Highly Reliable Flight Control Systems," The Charles Stark Draper Laboratory, Inc., Cambridge, Mass., 1975.
41. Van Trees, H.L., Detection, Estimation, and Modulation Theory, Part I: Detection, Estimation, and Linear Modulation Theory, John Wiley and Sons, Inc., New York, 1971.
42. Wernersson, Ake, "On Bayesian Estimators for Discrete Time Linear Systems with Markovian Parameters", Sixth Symp. on Nonlinear Estimation and Its Applications, San Diego, Calif., Sept. 15-17, 1975.
43. Bueno, R., Chow, E., Gershwin, S.B., and Willsky, A.S., "Research Status Report to NASA Langley Research Center: A Dual-Mode Generalized Likelihood Ratio Approach to Self-Reorganizing Digital Flight Control System Design", M.I.T. Electronic Systems Laboratory, Cambridge, Mass., Nov. 1975.
44. Van Trees, H.L., Detection, Estimation, and Modulation Theory, Part III: Radar-Sonar Signal Processing and Gaussian Signals in Noise, John Wiley and Sons, Inc., New York, 1971.
45. Doolin, B.F., "Reliability Issues for Future Aircraft," MIT-NASA/Ames Workshop on System Reliability Issues for Future Aircraft," M.I.T., Cambridge, Mass., August 18-20, 1975.
46. Taylor, L., "Active Control Aircraft Problems," MIT-NASA/Ames Workshop on System Reliability Issues for Future Aircraft," M.I.T., Cambridge, Mass., August 18-20, 1975.
47. Meyer, G., and Cicolani, L., "A Formal Structure for Advanced Automatic Flight-Control Systems", NASA TN D-7940, May 1975.
48. Zyweitz, Chr. and Schneider, B., Computer Application on ECG and VCG Analysis, North Holland, 1973.
49. Segall, A., "A Martingale Approach to Modeling, Estimation and Detection of Jump Processes", Ph.D. Dissertation, Stanford University, Stanford, Calif., August 1973.
50. Snyder, D.L., Random Point Processes, John Wiley and Sons, Inc., New York, 1975.
51. Athans, M., Whiting, R.H., and Gruber, M., "A Suboptimal Estimation Algorithm with Probabilistic Editing for False Measurements with Applications to Target Tracking with Wake Phenomena", Sixth Symp. on Nonlinear Estimation and Its Applications, San Diego, Calif., Sept. 15-17, 1975.

#### Acknowledgements

The author is in the debt of many colleagues for their comments during numerous discussions on the subject of failure detection. In particular, special thanks must go to Dr. R.C. Montgomery, Dr. A.K. Caglayan, Dr. D.B. Price, and Mr. J.L. Elliott of NASA Langley Research Center, Mr. B.F. Doolin of NASA Ames Research Center, Dr. H.L. Jones, Dr. B.S. Crawford, Dr. T.H. Kerr, Mr. J.H. Fagan, and Mr. W. O'Halloran of The Analytic Sciences Corporation, Dr. J.J. Deyst, Dr. D.E. Gustafson, Mr. J.C. Deckert, Dr. M. Desai, Dr. J.L. Speyer, and Mr. J.-Y. Wang of The Charles Stark Draper Laboratory, Dr. K.P. Dunn of the M.I.T. Lincoln Laboratory, and Prof. M. Athans, Dr. S.B. Gershwin, Mr. E. Chow, and Mr. R. Bueno of M.I.T.

This research was supported in part by NASA Ames Research Center under Grant NGL-22-009-124 and in part by NASA Langley Research Center under Grant NSG-1112.

A version of this paper appeared in the November 1976 issue of Automatica.

# CAST - A COMPLEMENTARY ANALYTIC-SIMULATIVE TECHNIQUE FOR MODELING COMPLEX, FAULT-TOLERANT COMPUTING SYSTEMS

by

R. B. Conn, P. M. Merryman, K. L. Whitelaw  
Ultrasystems, Inc.  
2400 Michelson Drive  
Irvine, California 92715  
USA

## SUMMARY

The complementary analytic-simulative technique (CAST) evolved as it became evident that neither analysis nor simulation alone could satisfy the evaluation requirements of complex computer systems. Analytic modeling provides flexibility and rapid, economical data generation. Simulation permits computer system details to be included easily, but extensive data generation is slow and expensive. CAST permits the user to obtain the best features of both analytic modeling and simulation. CAST is based on the desirability of accurately modeling complex computer systems, utilizing closed-form mathematical expressions for the computer system failure probability. This is achieved through the use of analytic models which utilize parameters determined both by simulation and engineering characterization of the computer system. This concept was evolved under a NASA Langley Research Center contract concerned with the reconfigurable computer systems (RCS) for commercial aircraft. The work was then continued for application to the Shuttle-orbiter data processing subsystem (DPS) on a NASA Johnson Space Center contract.

CAST has been used to determine the survivability of one of the early test configurations of the Space Shuttle Data Processing Subsystem (DPS). The DPS mission-critical survivability for a six-hour mission was determined to be 0.999863 for the Shuttle ALT baseline configuration. The analysis led to the evaluation of three selected options which identified two areas of possible improvement. The Shuttle work included: extending the GPC analytic model to include imperfect detectability; creating a new analytic model to handle configurations involving non-symmetrical interconnections; creating a new analytic model to handle combinations of dependent device sets (e.g., flight-critical bus and connected units); adding three routines to reflect transient recovery procedure differences; and developing a simulation for the flight-critical-bus partition.

CAST also has been applied to a number of example computer system configurations for avionics applications to provide insight into the important aspects of these configurations. The conclusions are based on a ten-hour flight and failure rates thought to be applicable to the off-the-shelf avionics computers studied. The reconfigurable computer systems were assumed to be composed of as many as five machines. It was determined that the greatest improvement in system survivability is obtained by increased redundancy. Each increment of redundancy decreases the 10-hour failure probability by approximately two orders of magnitude. The greatest failure probability decrease occurs when changing from triplex to quadruplex, e.g., a 200-fold improvement. Increasing redundancy also increases cost in terms of power, weight, and volume not only due to the added units but due also to the increased complexity of intercommunications modules, external electronics modules, and bus switches.

Increasing redundancy has diminishing returns if there are errors in permanent-recovery algorithm design. This error penalty becomes more severe with added redundancy. Using simpler recovery algorithms, i.e., those involving less RCS adaptivity, is a possible way of ensuring error-free recovery. However, the increase in failure probability for air-transport-type missions due to decreased adaptivity (e.g., not adapting the system down to one computer) is less than that caused by decreased redundancy or recoverability. Since redundancy has such a large effect on failure probability, external hardware should have an equivalent redundancy to prevent external failures from depressing the overall survivability.

The analytic modeling has resulted in the successful formulation of survivability expressions for an RCS composed of  $N$  computers, where  $N$  is any finite, positive integer. The models included explicit consideration of the components of coverage and the effects of transient faults. Hardware/software interactions are included implicitly.

The simulator portion of CAST is based on descriptions of the states the RCS could assume and the transitions between these states. Use of this approach results in a fault-driven simulation which minimizes the simulation cost. The RCS simulator requires that the fault environment, the configuration particulars, the system failure criteria, the software structure, the test features, and the recovery approaches be specified. The modeling parameters, plus other useful data such as state-transition statistics, result from the simulation.

The techniques reported here devote much attention to the modeling of transient faults. The results show that a knowledge of the transient environment results in effective transient recovery features. Underestimating transient duration results in many transients being recorded as permanent, while overestimating transient duration leaves the system unduly vulnerable to further faults.

## LIST OF SYMBOLS

$A_i$	Device type-A (ith unit)	$c_i$	Coverage, defined as the product of $u_i$ , $v_i$ , and $w_i$ ; i.e., the probability the system recovers given a fault occurs
$B_i$	Device type-B (ith unit)		
$C_i$	Controller i		
$D_T$	Transient duration	$l_i$	Transient leakage (probability of failure of transient recovery given that fault is transient)
$R_c(T)$	Reliability of the controller at time T	$u_i$	Detectability (probability fault is detected given fault occurs)
$S_i(T)$	Survivability of system at redundancy level i (probability that system functions correctly over a mission of length T)	$v_i$	Diagnosability (probability fault is isolated given fault is detected)
$S_i(T,A)$	Survivability of device A at redundancy level i	$w_i$	Recoverability (probability system recovers given fault is detected)
$S_i(T,B)$	Survivability of device B at redundancy level i		
$T$	Mission duration	$\lambda$	Permanent fault rate
		$\tau$	Transient fault rate

## 1. INTRODUCTION

As more and more demands and responsibility are placed on the computer in a flight control system, the need for computer fault tolerance has increased. As with any non-physical feature of an avionics subsystem, it is necessary to estimate various parameters before, during, and subsequent to the design process. Thus it is necessary to evaluate the fault tolerance of the computer subsystem in a flight control system.

This paper describes the first two phases of work being performed for the U.S. National Aeronautics and Space Administration (NASA). The approach described here is different from those taken in the past in that it is more comprehensive and detailed. The work is designed to produce usable tools--i.e., tools that can actually be put to work in the engineering design process of a flight control system.

The main body of the paper is divided into six sections. The first of these is intended to provide the reader with an overview of the concept before he becomes enmeshed in the mathematics and flow charts. The second section describes various fault-tolerance concepts so that the paper can be read without extensive perusal of references. Following these introductory sections, the analytic models and simulative models are described in the next two sections. The next section describes the applications of the technique which have been made to date. Finally, conclusions are presented.

## 2. THE COMPLEMENTARY ANALYTIC SIMULATIVE TECHNIQUE

The Complementary Analytic-Simulative Technique (CAST) evolved as a result of a study performed for NASA Langley Research Center. The objective of the study was to provide concepts and engineering data from which a highly reliable, fault-tolerant, reconfigurable computer system (RCS) for aircraft applications could be designed. For the purposes of the study, an RCS was defined to be a redundant configuration of off-the-shelf avionics computers which achieved fault tolerance through use of a variety of recovery techniques. A principal study goal was the development and application of reliability and fault-tolerance assessment techniques. Particular emphasis was placed on the needs of an all-digital, fly-by-wire control system appropriate for a passenger-carrying airplane. A representative set of results obtained from applying CAST to an RCS is shown in Figure 1.

This Complementary Analytic-Simulative Technique (CAST) evolved as it became evident that neither analysis nor simulation alone could satisfy all the RCS evaluation requirements. Analytic modeling provides flexibility and rapid, economical data generation. However, the solutions for some configurations are very cumbersome and, in certain cases, the mathematical model formulated is intractable. Simulation permits computer system details to be included easily, but data generation is slow and expensive. CAST permits the user to obtain the best features of both analytic modeling and simulation. When these two system evaluation approaches are combined, and are supplemented by an engineering characterization of the system, a very powerful technique results. The combination is illustrated in Figure 2.

The engineering characterization is performed to provide six categories of information to the analytic modeling and the simulation. These information categories are: (1) configuration particulars, (2) fault environment, (3) system failure criteria, (4) software structure, (5) recovery features, and (6) test features. The individual items in these six categories are shown in Figure 2. The following items are available as simulator outputs: (1) permanent-fault coverage, (2) transient-fault coverage, (3) detectability, (4) diagnosability, and (5) recoverability. The analytic modeling provides the following measures of fault tolerance: (1) computer system survivability (or failure probability), and (2) computer system reliability.



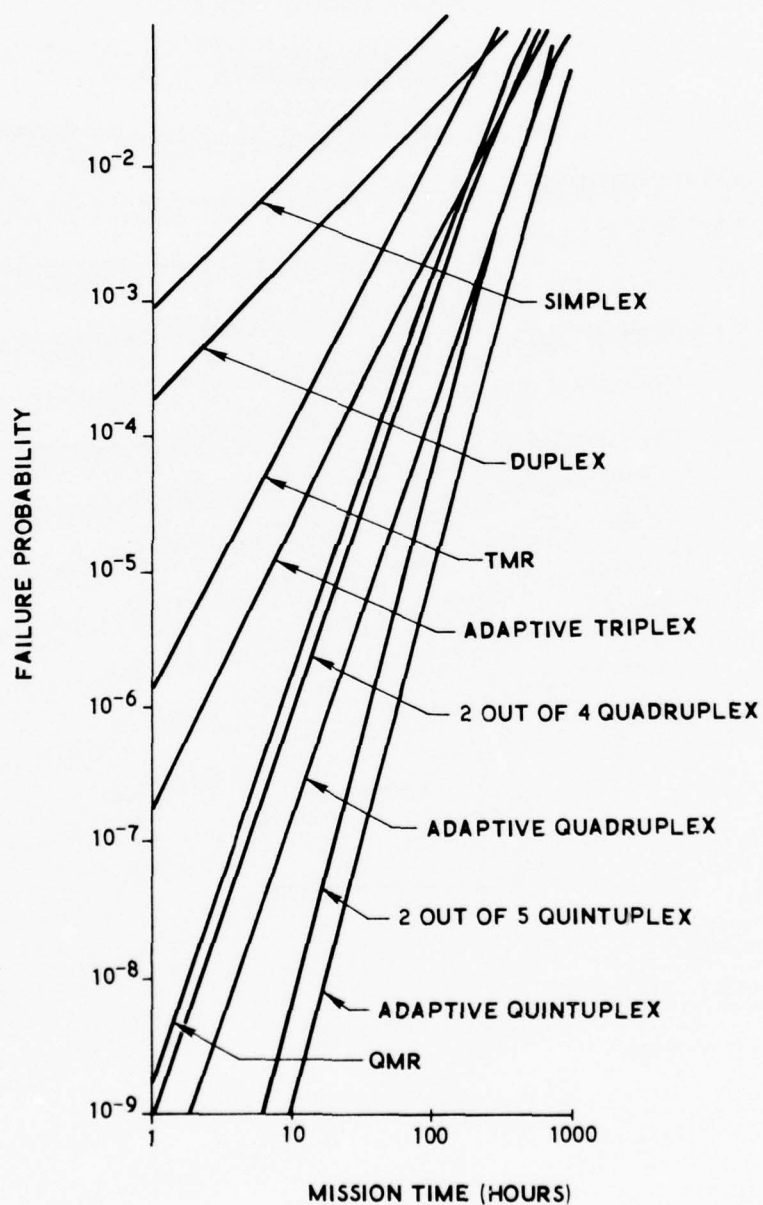


FIGURE 1 EFFECTS OF COMPUTER SYSTEM REDUNDANCY AND ADAPTABILITY ON FAILURE PROBABILITY



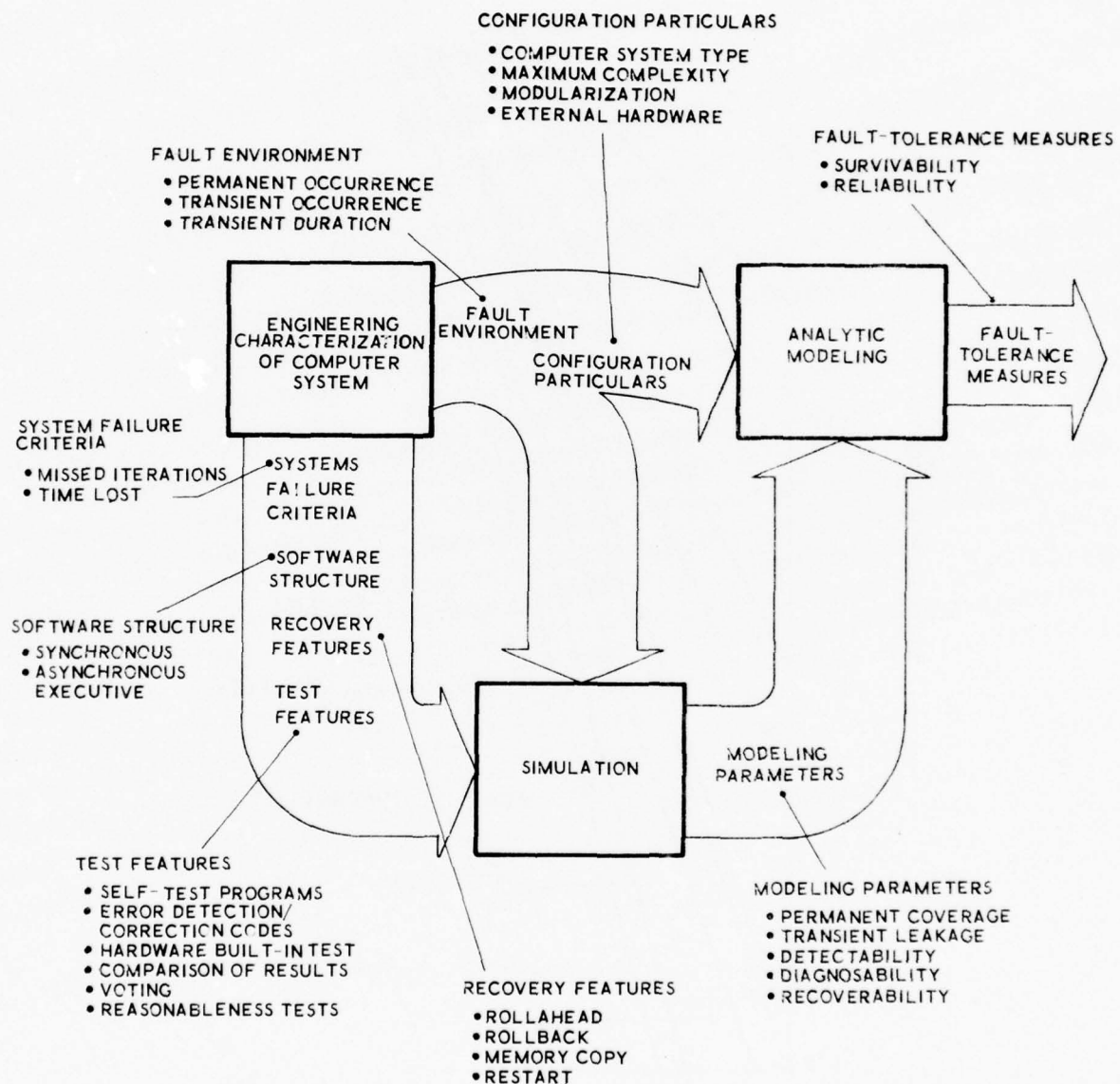


FIGURE 2 CAST ACTIVITY SEQUENCE AND INFORMATION FLOW

### 3. CONCEPTS OF FAULT TOLERANCE

There are two general methods of measuring the effectiveness of the fault tolerance. One way is to count the worst-case number of faults that a system will tolerate before failure. An example of this is a fail operational/fail safe (FO/FS) specification. The specification of the number of faults to be tolerated gives the user a feeling for the redundancy employed and gives him the reassurance that a fault will not bring down the entire system. But a measure is needed to specify the probability that the system will not fail over a time span. This other measure is generally called the mission success probability. The mission success probability that is most commonly used is the reliability. This is the probability that at least one set of functioning hardware survives the entire mission. Reliability considers only permanent hardware faults. Survivability is the measure we shall consider here. It considers the effects of transient faults as well as permanents in its results, and is the result of the application of CAST.

#### Configurations Under Consideration

CAST is applicable to multicomputer systems which are formed by interconnecting identical computers into a symmetrical fault-tolerant configuration. These computers perform identical functions in synchronism so that valid results are available in case of an erroneous computation. The failure of an individual computer is handled by switching it out of the system and continuing system operation with one less computer. Ideally, the system could lose, because of component failures, all but one computer and still perform properly. However, unless the mechanisms for fault detection, fault location, and system recovery are perfect, the system may not recover from a fault even though adequate redundancy is available. It is important to note that the input/output bus system and devices also must be protected by redundancy in order to guarantee the critical data processing tasks are performed. The important features of a fault-tolerant multicomputer system are: (a) synchronization of inputs, outputs and task performance; (b) fault detection and diagnosis; and (c) transient and permanent recovery.

#### Fault-Handling Methods

Fault detection is the antecedent of actions necessary to recover from a fault. In a multicomputer system, fault detection is accomplished either by cooperative action between computers or by self-detection methods. Cooperative action includes such methods as voting or comparison of results. Voting may be a compilation of agreements or disagreements of comparisons between three or more machines, or it may be performed by a hardware voter. Location of the faulty machine is immediate from detection by voting, but detection by comparison between two machines (as in duplex) still leaves the faulty machine to be resolved by self-test. Self-detection methods include hardware built-in test equipment or software diagnostics. Examples of hardware built-in test in current computers include such things as memory parity, watchdog timer, illegal operation code, or power failure. These methods seldom achieve high coverage by themselves, but in conjunction with software diagnostics they are the only method of resolving the faulty computer after fault detection by comparison between results from two remaining computers.

Faults may be either transient or permanent. A permanent fault is a hardware failure that prevents the computer from functioning. Recovery may be effected by ignoring the faulty machine, removing its power, masking by the voters, or by other methods of removal from the redundant set. Before removal we must first determine whether the fault was caused by a transient, and if it was, restore the faulty machine. The method of transient determination is usually to attempt a recovery sequence. If the error has been removed, the fault will not be redetected. Transient faults are caused by a temporary malfunction that lasts for a relatively short duration. Longer duration faults can last sufficiently long to be recorded as permanent failures. Shorter transients, however, may leave one or more errors in program and/or data during their stay. Transient recovery methods are designed to correct these transient-caused errors. Destructive readout memories are particularly vulnerable to transients during the restore cycle. Data may be altered during a read cycle. Data may be altered by transients in the CPU or memory. Rollahead and rollback are two methods of recovering from errors in the current data in order to continue computation. Rollback is a self-recovery scheme where the computational segment in which the error was detected is repeated with the **previous** data state. Rollahead is a cooperative recovery method where the valid current results are passed from the good computer to the disagreeing computer and computation resumes. Two or more good computers are required for rollahead while rollback is independent of the redundancy level. Neither of these methods correct program memory faults. There is a cooperative method called memory copy that accomplishes this. Here the memory contents of the good computers are voted into the disagreeing machine at a low duty cycle. This allows real-time computations to continue. After the memory copy is completed, a rollahead is performed to place the recovered computer into synchronization. Memory copy recovery leaves the system with degraded redundancy for a period of time, but it corrects many transients that would otherwise be mistakenly labeled permanent.

### 4. CAST APPLICATIONS

CAST has been applied to the space Shuttle-orbiter data processing system (DPS) for the approach and landing test (ALT) configuration to obtain mission survivability and evaluate possible design modifications. Before describing the applications, a brief description of the DPS will be presented.

### Shuttle Data Processing System (ALT)

The Shuttle DPS is composed of five identical general-purpose computers (GPCs) connected with each other and the peripheral devices by a system of redundant serial buses. In the ALT configuration the fifth GPC is set aside as backup flight control for the landing tests and is not included in the modeling. All functions that are not related to the return flight are not in the ALT configuration and are hence not modeled. The resulting ALT configuration is shown in Figure 3. As shown in the figure,

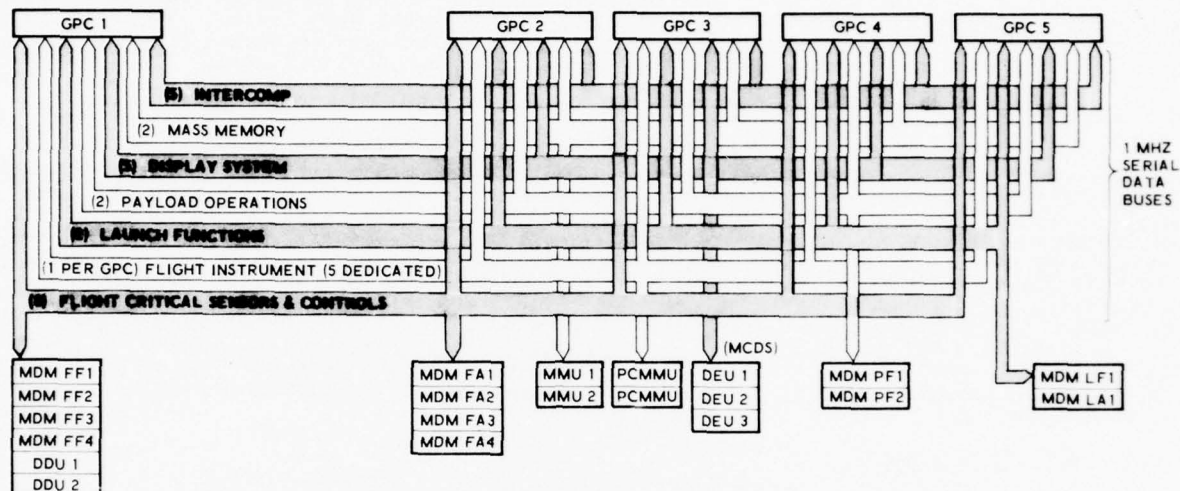


FIGURE 3 SHUTTLE ORBITER COMPUTER SYSTEM BLOCK DIAGRAM (ALT)

communication among the GPCs, and between the GPCs and/or the peripheral devices is effected through use of seven groups of buses. The number of buses in each group is shown on the figure. Each of these buses is a one-megahertz, serial bus. Communication between units on a bus is accomplished through use of command words, command data words, and response data words. Each GPC is composed of a central processing unit (CPU), memory, and an input/output processor (IOP). All information transfers to and from the GPCs are handled through the IOP. Software control is used to instruct each bus within a data-bus group whether it is to operate in the command or listen mode. When operating in the command mode, data requests and commands are sent to the peripheral equipment and the data is then supplied over the same bus. When in the listen mode, data are only received on the bus.

The bus configuration allows each computer to have access to all flight-critical data received or transmitted by the other computers. Each of the redundant subsystems is connected to a different bus. Hence for data input, a different computer requests data from each of the subsystems. The requested data are then available to all other computers. Thus identical input data are available to each computer in the DPS. For data output, since each channel of the actuator subsystem is connected to a different bus of the group, a different computer transmits command data to each of the voting actuator channels. As a result of the bus-computer interconnections, each computer can monitor the command data sent out by each of the other computers. When data is to be transferred between computers, each computer communicates with all other computers through the intercomputer communication (ICC) buses. Only the GPCs are connected to the ICC buses. In order to avoid data skew of either inputs or outputs, synchronization is accomplished in the DPS through use of intercomputer discrete signals and synchronization software. Sensors and actuators are connected to the appropriate bus through multiplex-demultiplex (MDM) units. Analog flight display units are connected to their bus through display driver units (DDU), while the multifunction CRT display system (MCDS) is connected through display electronic units (DEU). The pulse code modulation master units (PCMMU) are connected directly to the GPCs by dedicated buses. The mass memory, launch, and payload buses are not applicable to ALT but are included in the figure for completeness.

Fault detection in the GPCs is accomplished by five methods: compare-word sum check, bus channel timeout test, built-in test equipment, watchdog timer, and self-test programs. The compare-word sum check involves summing critical GPC actuator-command outputs, and each GPC comparing its sum with that of the others. This check is performed each computation cycle. This comparison is performed by use of the Fault Detection Identification Program. If the difference is greater than that allowable and has occurred the maximum permissible number of times, then the fail-discrete of the faulty GPC is set. There are two recovery approaches available in the Shuttle GPC configuration. The first of these is one in which the crew identifies a failed GPC through use of the "failed-discrete" and may either switch out the failed machine or try an initial program load (IPL). The IPL approach is used when there is reason to believe that a transient fault has been



experienced. The second recovery approach is to crew-enable inhibition of transmission of outputs from the failed GPC. This inhibition is accomplished automatically once it has been enabled by the crew. It should be noted that restoration of a GPC that may have suffered a transient is not attempted during the action portion of ALT. This is because of the stringent recovery time constraints and the fact that restoring and adding a computer to the redundant set during time-critical mission phases requires a significant amount of computer memory and time and introduces greater than desirable operational complication. Fault detection in the peripheral units of the DPS is accomplished by a combination of BITE and GPC-supervised tests. The recovery approach used depends upon the particular unit.

#### Shuttle Data Processing System Partitioning

The Shuttle DPS is a complex system. It consists of extensive peripheral devices for the control of the vehicle and its payload through launch, orbit maneuvers, and the return flight. The exact modeling of this complex system as a whole is not feasible because of the mathematical complexity necessary to account for unit interactions. A systematic method of reducing the problem to solvable pieces is required. Such a method is the partitioning of the system into statistically independent module sets. By independence of module sets, we mean independence with respect to the impact of faults from one set to the other. A definition of independence is as follows: Given a collection of module sets, the sets are independent of each other if a faulty module within one set does not incapacitate modules within any other set. However, within each independent module set, a failure of one module type has an effect on other module types. For example, a CPU fault would cause its IOP to not function properly, and an MDM failure would prevent access to the devices it services. Having defined the independent partitions, the survivability of each partition may be determined independently and the system survivability is the product of the survivabilities of the partitions.

The first-cut partitions are along the lines of the bus groups. These groups are: the four general-purpose computers (GPC); the flight-critical buses and connected equipment (FCB); the display equipment and their buses (MCDS); and the flight instrumentation and buses (PCM). A failure of one of these groups has a different impact on the Shuttle mission depending on the group. There are two levels of failure criticality: safety-critical and mission-critical. Safety-critical failures threaten the Shuttle vehicle and the lives of the crew, while mission-critical failures affect the accomplishment of the mission. A bus group falls into one of these two categories. The safety-critical partitions for ALT are: the GPCs, the flight-critical bus group, and the MCDS. A safety-critical failure is also mission-critical since a lost vehicle implies an unsuccessful mission. Therefore, safety-critical partitions are also mission-critical. The flight instrumentation is mission-critical.

The flight-critical bus system consists of 8 buses connected to 4 forward MDMs, 4 aft MDMs, and 2 DDUs. Failure in one of these module groups does not affect the other module groups. Bus failures do affect more than one module group, but the bus failure rate is very small compared to those of the modules. Because it is small, the bus failure rate is very small compared to those of the modules. Because it is small, the bus failure rate can be included with each of the module groups with a very small resultant error. The result is a slightly pessimistic estimation of the survivability. Therefore, the forward MDMs, aft MDMs, and DDUs, with the buses attached to each, constitute three more partitions.

An additional aspect of complexity is the extensive input/output network. Additional analytic models and extensive simulator expansions are required to model these. An example of this complexity is the MCDS. Here there are two keyboards serving three display units with an unusual switching arrangement. Another area is the flight-critical bus. Here the MDMs or DDUs serve as peripheral controllers serving several devices. A device failure does not incapacitate other devices. However, an MDM or DDU failure incapacitates all devices served by it.

#### Shuttle (ALT) Results

In order to make mission success probability calculations using the analytic models, it is necessary to obtain, by simulation, values for the various parameters required in these models. The approach taken was to obtain a baseline set of parameters and then vary these parameters to reflect the several options investigated. Approximately 200 parameters for both the models and simulator are required to be specified. The survivability results for the safety-critical partitions, as well as the overall safety-critical and mission-critical results are given in Figure 4. Failure probability, which is one minus survivability, is shown on a logarithmic scale as a function of time. It shows that the forward flight-critical MDMs and devices are the greatest contributor to the overall failure probability. This is due to lower coverage on the devices than the GPCs, the devices' relatively high failure rates and a lower device redundancy.

Figure 5 graphically illustrates how increased redundancy improves failure probability. The figure is a log-log plot with mission times from one to 1000 hours and failure probabilities from  $10^{-9}$  to .1. The higher the level of redundancy, the better the failure probability prediction as one would expect. The GPCs have a higher redundancy level than the flight-critical devices, so that their survivability prediction is greater. But any additional increases in GPC redundancy gain nothing unless the other system components are also increased in redundancy. Figure 6 demonstrates how improvements in detectability, diagnosability, and recoverability improve the mission failure probability. The figure



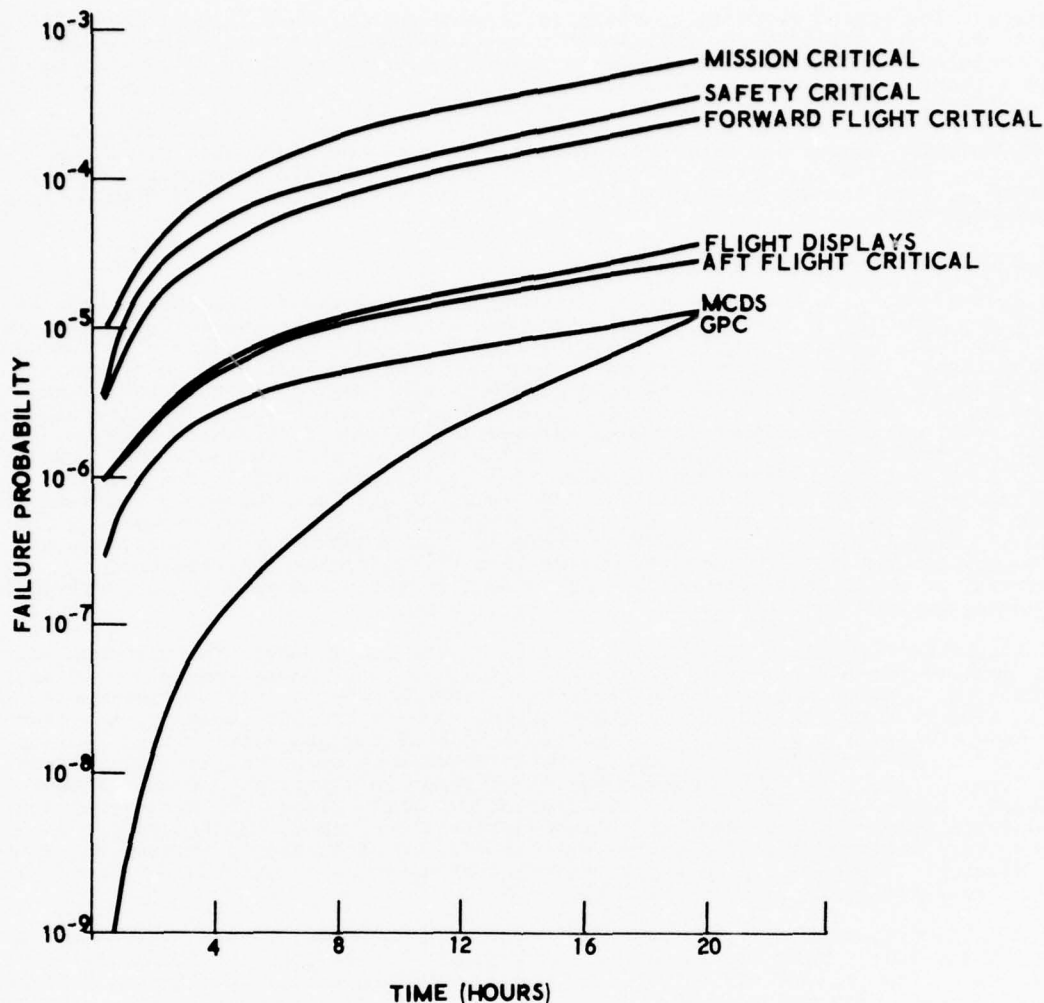


FIGURE 4 SHUTTLE BASELINE FAILURE PROBABILITIES

shows how the profile of failure probability versus time is affected by different recoverability values for the four computers. The straight-line portion is for the perfect coverage case. For imperfect recoverability, the failure probability for early mission times is dominated by coverage until it "catches up" with the perfect coverage case. It does not immediately join the perfect coverage failure probability curve, but approaches it as an asymptote. For shorter mission times, such as in the Shuttle ALT, there is much to be gained by improving the coverage components. Adaptive configurations can adjust to a new fault-tolerant scheme when units have been recorded as faulty. Non-adaptive configurations are duplex, TMR, QMR (three-out-of-five vote), and two-out-of-n votes ( $n = 4, 5$ ). Because a two-out-of-five vote is capable of recovering from a fault with three GPCs remaining, it is, in a sense, more adaptable than QMR which fails after a fault with three computers. Quintuplex, quadruplex, and triplex are adaptive cases where voting is the prime method of fault detection and diagnosis with three or more fault-free computers. The residual duplex mode is entered when two computers remain. The resulting failure probabilities versus time are shown in Figure 7.

**Transient Recovery Effectiveness.** The section on the concepts of fault tolerance discussed some transient fault recovery methods. Here we show what improvements are possible from the present Shuttle recovery method. The Shuttle transient recovery method, which is a delay before attempting a permanent-fault recovery, is quite effective for transient faults occurring external to the GPCs. This is due to the filtering of the processing algorithms and the slow response time of the actuators and displays. This recovery method is not as effective for transients within the GPC. It is easy for a program to be altered by a memory transient during a restore cycle. Also, CPU and IOP transients can alter data. Thus, a GPC can be left with a "permanent" fault actually resulting from a transient. The three alternate transient-fault recovery options studied here are rollback, rollahead, and a combination of rollahead and memory copy. As an instructive additional exercise, we examine the case where the GPC memory is non-destructive readout (NDRO). This is not the case for Shuttle, but the results are interesting. Rollback is the procedure where the current program segment is rerun following fault detection. Rollahead is the procedure

where the fault-free GPCs pass the current machine-state and data points to the indicated faulty machine and continue computation. Memory copy is the procedure where the contents of the memories of the good GPCs are passed to the faulty GPC at a low duty cycle on a cycle-stealing basis. Memory copy is followed by a rollahead after completion to bring the faulty GPC on line. The results are given in Figure 8 as a plot of six-hour failure probability versus transient fault rate. Delay recovery exhibits the greatest sensitivity to transient faults. Rollahead is not as effective as rollback because it is not applicable in duplex. Memory copy is the best recovery method, but using an NDRO memory (with rollback or rollahead) does the best job of reducing the effects of transient faults. This is because there is no restore cycle in an NDRO memory and it will not suffer transient damage during a read operation.

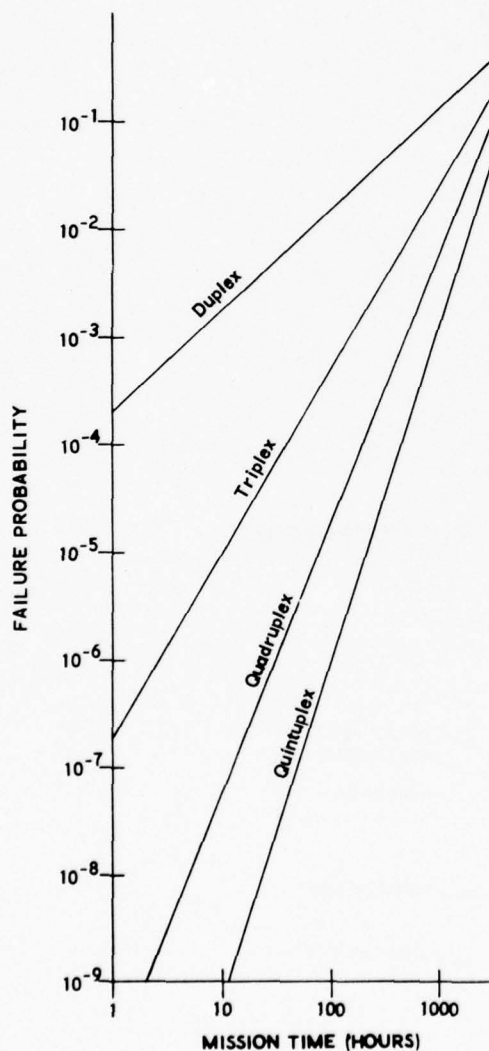


FIGURE 5 COMPARISON OF REDUNDANCY LEVELS

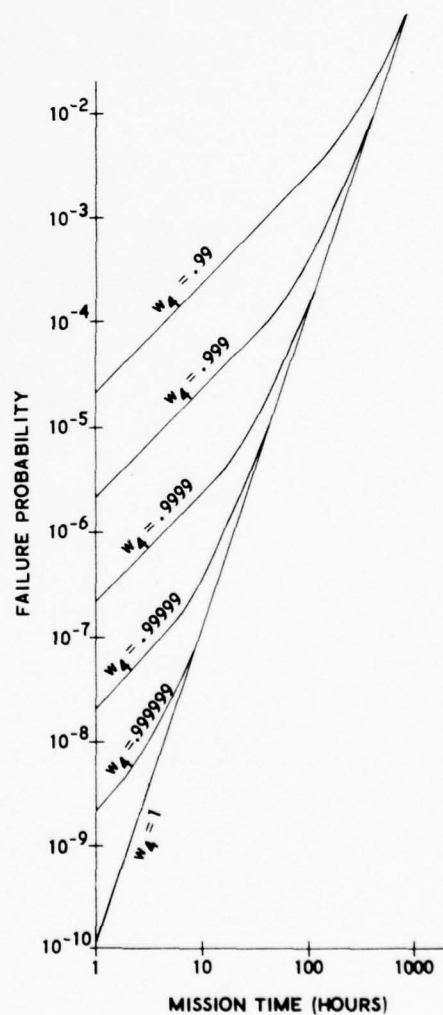


FIGURE 6 RECOVERABILITY EFFECT ON GPC FAILURE PROBABILITY

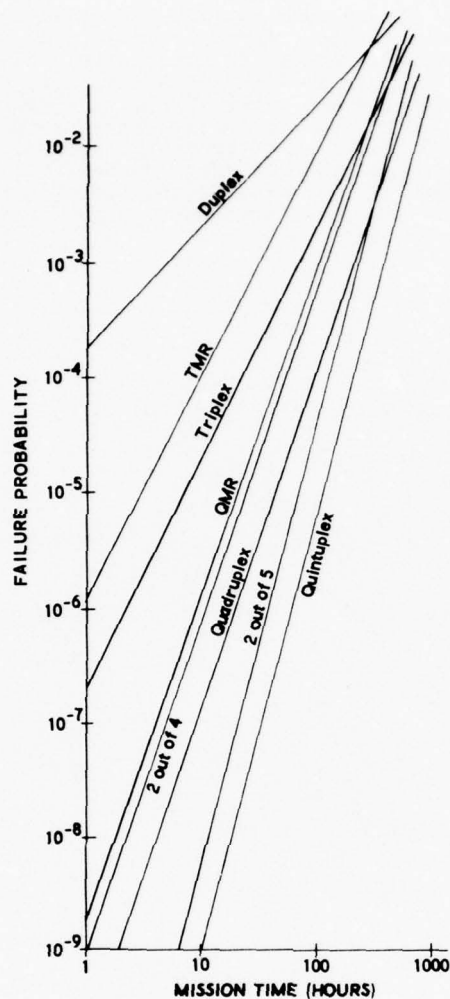


FIGURE 7 EFFECTS OF VARIOUS ADAPTATION SCHEMES

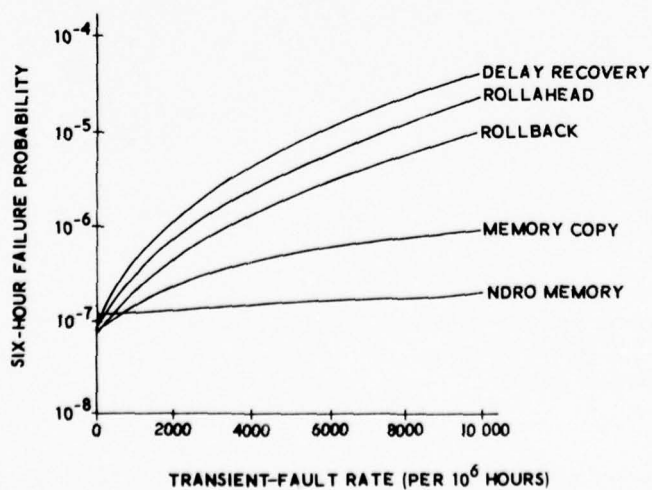


FIGURE 8 EFFECTIVENESS OF TRANSIENT RECOVERY METHODS OVER A RANGE OF TRANSIENT-FAULT ENVIRONMENTS

## 5. ANALYTIC MODELING

The generation of analytic models is a five-step process. The steps to be taken to find expressions for survivability is as follows:

- a. Characterize transient faults;
- b. Characterize the effectiveness of fault coverage;
- c. Formulate a fault occurrence/recovery status state diagram;
- d. Form equations;
- e. Solve.

The methodology will be illustrated by the simple example of a triple-modular-redundant (TMR) system. After the initial consideration of steps a and b above, the results will hold for the more complex configurations that will follow.

### Transient Fault Characterization

A transient fault is a fault that disappears some time after its arrival. During its stay it alters the contents of registers and/or memory and/or disrupts the normal sequence of program execution. We recover from a transient that has passed by restoring altered data and/or program and by bringing the recovering computer into synchronization with the fault-free computers. We will characterize transient faults by their arrival and their duration. In our arrival characterization, we make the assumption that transient faults arrive with an average rate  $\tau$  that is constant during a mission. With the constant rate over time, the probability of the arrival of a transient fault in a small interval of time,  $dt$ , is  $\tau dt$ . It is well-known that under these conditions (see Refs. 1 and 2) the probability of exactly  $k$  transient fault arrivals between 0 and  $t$  obeys a Poisson probability law. That is

$$\text{Pr} [k \text{ arrivals in } (0,t)] = e^{-\tau t} \frac{(\tau t)^k}{k!}$$

If we let  $k = 0$ , we have

$$\text{Pr} [\text{No transients in } (0,t)] = e^{-\tau t}$$

which is analogous to the characterization of permanent fault and leads to an analogy between  $\lambda$ , the permanent fault rate and  $\tau$ . We assume transient faults have a definite duration. There is a dilemma concerning the probability density function of the duration: We could be of the opinion that short transients are much more likely than long transients which would lead us to an exponential density as a mathematically tractable approximation. We could also be of the opinion that there is definite mean duration with an associated spread which would lead us to the gamma, normal or Weibull densities as an approximation. We could also say that transient faults are caused by several sources, each source with a different average duration. But there are more sources with a small duration than with a large duration. In this case, the composite density function of all the durations could be a "lumpy" exponential.

### Fault Recovery Process

When faults occur in a fault-tolerant computer system, certain events must take place for the eventual recovery and resumption of computations. This sequence is called the fault-recovery process. After fault occurrence, it must be detected. After detection, the fault must be correctly located (diagnosis). And after diagnosis, an appropriate recovery action must take place. This sequence of detection, diagnosis, and recovery is complicated when transient fault recovery is attempted. The sequence of events that occur is shown in Figure 9. The fault is first detected; then a recovery delay may be invoked to allow the transient phenomenon to subside. (Transient recovery methods need no diagnostic procedure.) After the delay, a recovery mode is entered. First, transient recovery is attempted. If it is successful, recovery is complete without redundancy degradation. If it fails a permanent fault is assumed, and another level of diagnosis and recovery is entered upon.

Detection, diagnosis and recovery each have two parameters associated with them: A probability of success, and a time to completion. Time is a simulator function that is used, in conjunction with other parameters, to provide the success probabilities to the models. These success probabilities are detectability,  $u$ ; diagnosability,  $v$ ; and recoverability,  $w$ ; and are defined as:

$$\begin{aligned} u &\triangleq \text{Pr} [\text{Fault is detected} \mid \text{fault occurs}]; \\ v &\triangleq \text{Pr} [\text{Fault is located} \mid \text{fault detected}]; \\ w &\triangleq \text{Pr} [\text{System recovers} \mid \text{fault is located}]. \end{aligned}$$

The product of  $u$ ,  $v$ , and  $w$

$$uvw = c = \text{Pr} [\text{System recovers} \mid \text{fault occurs}]$$

\* means "equal by definition."



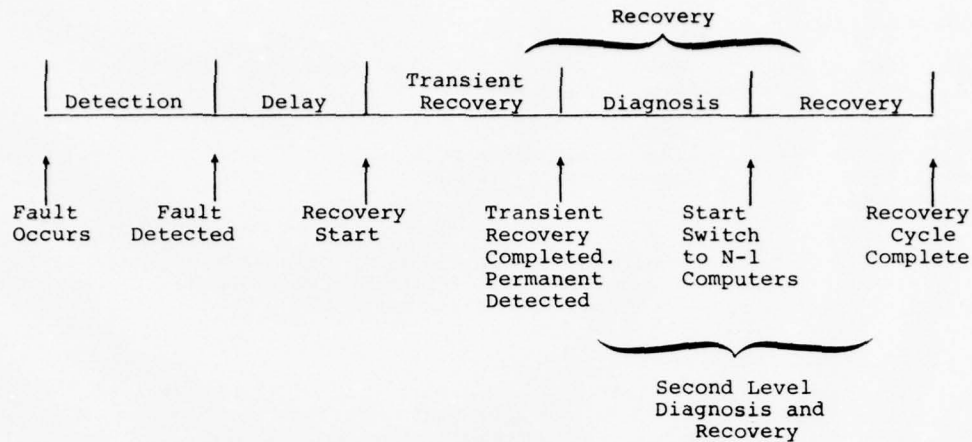


FIGURE 9 FAULT RECOVERY SEQUENCE

is the well-known coverage parameter (see Ref. 3). Previous models include this parameter, but do not break it into its component parts. These parameters will be given subscripts in the sequel. The value of the subscript represents the number of fault-free units remaining, and the parameters' applicability with that level of residual (or original) redundancy.

#### State Diagram for a TMR Configuration

We are now prepared to analyze a TMR system. We begin by drawing a fault occurrence/recovery status state diagram. This diagram represents the fault/recovery status of the system. Each state represents the number of fault-free units in the system and the level of fault recovery being undertaken. The transitions between states represent the occurrence of status-changing events. The events are random in general so that the state diagram is probabilistic in nature. Such a state diagram is illustrated in Figure 10. From the no-faults state a transient fault moves us into the transient recovery state. From the transient recovery state one of three things may occur:

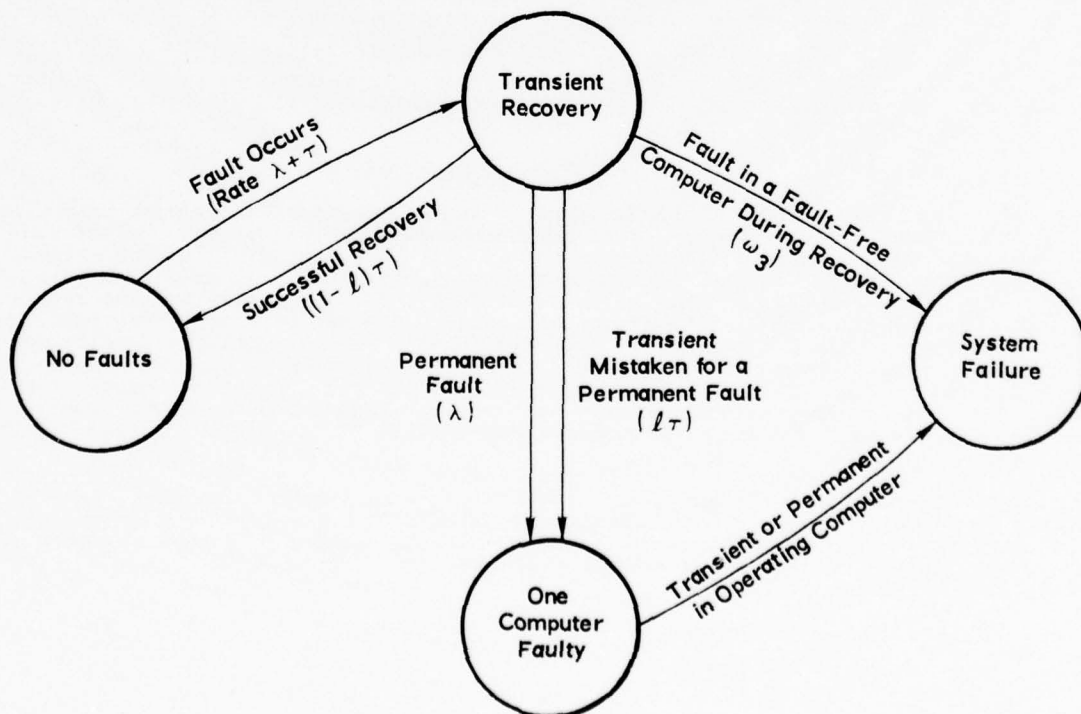


FIGURE 10 FAULT RECOVERY STATE DIAGRAM OF A TMR CONFIGURATION

- a. Successful recovery--go to No-Faults;
- b. Transient mistaken for a permanent--go to One Computer Faulty;
- c. Fault in a previously fault-free computer during recovery--go to System Failure.

A permanent fault will certainly be interpreted by the recovery procedure to be permanent. Therefore, for analysis purposes, a permanent fault in the no-faults state moves us to the one computer faulty state without passing through the transient recovery state. If a fault occurs in one of the two remaining computers in the one computer faulty state, the system fails. In TMR without transient recovery, the first fault is masked by the voter. Therefore,  $u_3$ ,  $v_3$ , and  $w_3$  are 1. In the transient recovery TMR (enhanced TMR),  $u_3$  and  $v_3$  are 1, but  $w_3$  is not quite 1 because a fault may strike down a good computer during recovery. However, it is very close to 1 (on the order of  $1-10^{-9}$ ).

#### Formulation of Equations

The first consideration before developing equations is to define the parameters to be used. They are as follows:

- $\lambda \triangleq$  Permanent Fault Rate;  
 $\tau \triangleq$  Transient Fault Rate;  
 $l \triangleq$  Pr [Transient Fault is interpreted as a Permanent];  
 $T \triangleq$  Mission Time;  
 $\sigma \triangleq \lambda + l\tau$ .

The quantity  $l$  is called the transient leakage and is the probability that the transient recovery procedure fails given a transient occurs. We also need the recoverability parameter,  $w_3$ , with a redundancy level of 3. The quantity  $\sigma$  is the rate of permanents and leaky transients. The TMR survivability is the sum of two mutually exclusive quantities. The first quantity is the probability there are no permanents or unrecovered transients during the mission. The second is the probability that a permanent or leaky transient occurs, and the system survives the remainder of the mission. Using these two quantities, the survivability is formulated as:

$$S(T) = e^{-3\sigma T} + \int_0^T \text{Pr} [\text{No permanents or leaky transients } \epsilon(0,t); \text{ a permanent or a leaky transient that is recovered at } t; \text{ and no transients or permanents in the two remaining computers } \epsilon(t,t+T)] dt$$

which becomes

$$S(T) = e^{-3\sigma T} + \int_0^T e^{-3\sigma t} 3\sigma w_3 e^{-2\sigma_t(T-t)} dt$$

and has a solution

$$S(T) = e^{-3\sigma T} + \frac{3\sigma w_3}{3\sigma - \sigma_t} \left[ e^{-2\sigma_t T} - e^{-3\sigma T} \right]$$

where  $\sigma_t \triangleq \lambda + \tau$ .

#### Modeling a Duplex Configuration

The duplex configuration is a step up in complexity from the TMR because a determination of the faulty computer is required. Fault detection is accomplished by results comparison between the two computers. Transient fault recovery is accomplished by rollback. If rollback fails, the faulty computer must be diagnosed by self-test methods. After diagnosis the faulty computer must not interfere with the operation of the good computer. Detectability is given a value of 1 while  $v_2$  and  $w_2$  lie between 0 and 1. In simplex, transient recovery is possible by using built-in test hardware to detect a fault and rollback to correct the transient error. The definition of the following parameters illustrate further the subscript method of specifying current redundancy.

- $l_2 \triangleq$  Pr {Transient mistaken to be permanent while in Duplex | Transient occurs}  
 $v_2 w_2 \triangleq$  Pr {Successful adaptation to Simplex | Permanent or Leaky Transient occurs}  
 $l_1 \triangleq$  Pr {Transient mistaken to be permanent while in Simplex | Transient occurs}  
 $l_2 \triangleq \lambda + l_2 \tau$

The quantity  $l_2$  is the transient leakage in duplex. The quantity  $v_2 w_2$  is the product of

the diagnosability  $v_2$  and the recoverability  $w_2$ . The quantity  $\sigma_2$  is the average rate of occurrence of permanents and leaky transients.

The fault occurrence/recovery state diagram of our duplex configuration is shown in Figure 11. From the no-faults state a transient will send us the rollback state where a rollback is attempted. It is successful with probability  $1-l_2$ . If the rollback is not successful, then the fault is taken to be a permanent from where diagnosis and recovery is initiated. If the fault is permanent, then it is taken to be permanent with probability 1. In diagnosis and recovery, a recovery to simplex is achieved with probability  $v_2 w_2$ . In simplex, only the rollback is possible.

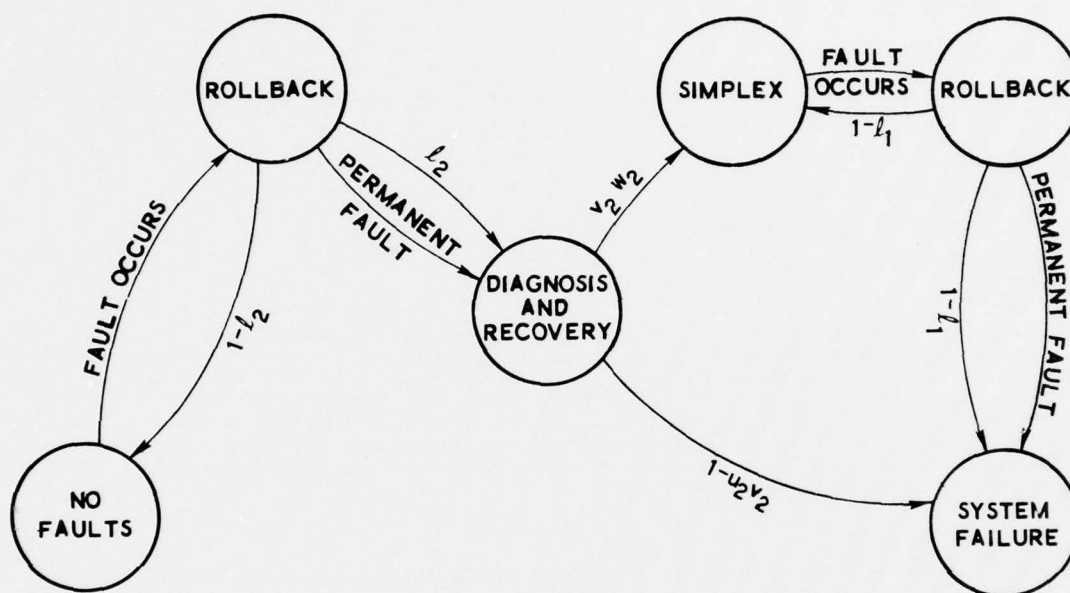


FIGURE 11 FAULT OCCURRENCE/RECOVERY STATUS STATE DIAGRAM FOR A DUPLEX CONFIGURATION

The survivability equation that may be formulated from this diagram is the sum of probabilities of two mutually exclusive events: The system has no permanents or leaky transients. And the system has a diagnosed and recovered permanent or leaky transient and survives the remainder of the mission in simplex. The duplex survivability,  $S_2$ , then becomes

$$S_2(T) = e^{-2\sigma_2 T} + \int_0^T e^{-2\sigma_2 t} 2v_2 w_2 \sigma_2 S_1(T-t) dt$$

where the simplex survivability is  $S_1(T) = \exp[-\sigma_1 T]$ . The solution to  $S_2$  is

$$S_2(T) = e^{-2\sigma_2 T} + \frac{2v_2 w_2 \sigma_2}{2\sigma_2 - \sigma_1} \left[ e^{-\sigma_1 T} - e^{-2\sigma_2 T} \right]$$

The integral equation for  $S_2$  suggests a recursive definition for  $S_N$  could be produced where 2 is replaced by  $N$  and 1 becomes  $N-1$ .

#### General Computer Configurations

The possibility of generalizing from the duplex configuration was suggested in the previous section. The analysis results in a recursive convolution integral equation that has a recursively defined solution. This generalization would be applicable to all configurations that undergo successive redundancy degradations from  $N$  to  $M$  computers before failure (N-M-1) and to those module types (such as computers) that a single failure disarms the entire module.

The state diagram of Figure 12 shows the sequence of events when  $N$  redundant computers are undergoing faults. We begin at time  $T=0$  in the  $N$  fault-free modules state and find the probability of the module set failing as a function of time. Faults occur at a rate  $\sigma$ , the sum of the permanent and transient fault rates. After a fault occurs, we move to

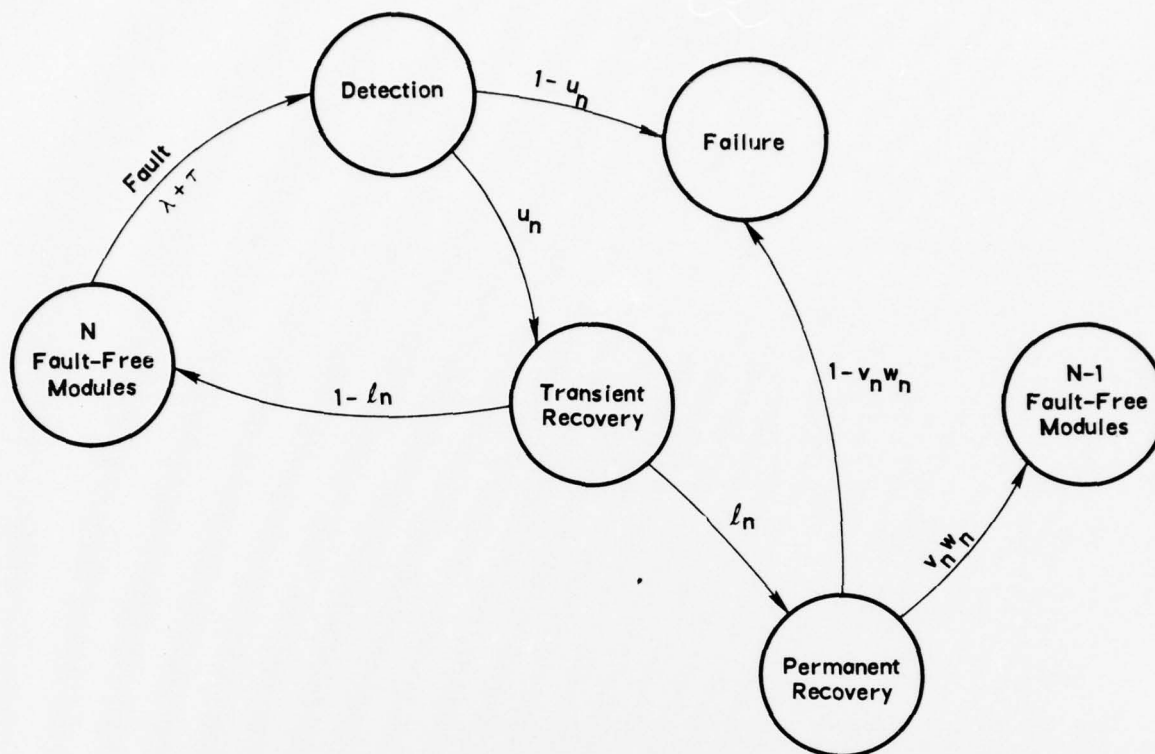


FIGURE 12 FAULT OCCURRENCE/RECOVERY STATUS STATE DIAGRAM

the detection state. With probability  $u_N$ , the detectability, the fault is detected, and we move to the transient recovery state. Failure to detect the fault is assumed to pollute the system with errors resulting in a system failure. After detection, a transient recovery is attempted. If transient recovery is successful, the module set is restored to  $N$  working units. Transient recovery is unsuccessful if the fault is permanent or with probability  $l_N$  (transient leakage) if the fault is transient. The unsuccessful transient recovery leads to a permanent recovery procedure where the module set redundancy is reduced by one. Failure of permanent recovery results in system failure.

In this model provision for non-unity detectability is added. This results in a change in definition of  $\sigma_N$  to

$$\sigma_N = u_N (\lambda + l_N \tau) + (1 - u_N) (\lambda + \tau)$$

For most practical values of  $u_N$  and  $\tau$ , this will not have a large effect, but it could have significance in some circumstances. We will also need coverage,  $c_N = u_N v_N w_N$  in our model derivation. The survivability equations for the general case are found by adding the probabilities of two mutually exclusive events, as before. The two events are: (a) There is no degradation of redundancy during the mission; and (b) there is a degradation to  $N-1$  computers, but the system survives the remainder of the mission. The survivability equation becomes

$$S_N(T) = e^{-N\sigma_N T} + N c_N \int_0^T e^{-N\sigma_N t} S_{N-1}(T-t) dt$$

whose solution is

$$S_N(T) = \sum_{k=1}^N \alpha_{Nk} e^{-k\sigma_k T}$$

where



$$\alpha_{Nk} = \frac{N C_N \sigma_N \alpha_{N-1,k}}{N \delta_{N-k} \delta_k} \quad k = 1, \dots, N-1$$

$$\alpha_{NN} = 1 - \sum_{k=1}^{N-1} \alpha_{Nk}$$

The solution is derived by assuming the sum of exponentials and substituting for  $S_{N-1}(T)$  and integrating.

#### A Model for the Shuttle Display System

In the Shuttle-orbiter data processing system, the operator's data entry and display system is known as the multifunction computer display system (MCDS). A special model was developed for the MCDS because of the peculiar interconnections of its redundant component modules. It consists of the display electronics unit (DEU), display unit (DU) and keyboard (KB). The DU is dedicated to the DDU, so we consider it a part of the DEU for analysis purposes. There are two KBs connected to three DEUs by a switching arrangement. The switching allows three configurations as follows:

1. KB A  $\longleftrightarrow$  DEU A  
KB B  $\longleftrightarrow$  DEU B
2. KB A  $\longleftrightarrow$  DEU A  
KB B  $\longleftrightarrow$  DEU C
3. KB A  $\longleftrightarrow$  DEU C  
KB B  $\longleftrightarrow$  DEU B

This connection arrangement is illustrated in Figure 13. The fault occurrence/recovery status state diagram is given in Figure 14. The detection, diagnosis, and recovery states are omitted in the diagram for clarity. At the beginning of the mission, the MCDS is in the no-faults state. If a keyboard fails, one of the DEUs will be permanently deprived of a keyboard. The mission continues with a simplex keyboard and duplex DEUs. If DEU C fails, then KB A will be dedicated to DEU A, and KB B will be dedicated to DEU B for the remainder of the mission. If DEU A or B fails first, then one KB is dedicated to DEU C while the other may be connected to either DEU C or B (we assume A was the failed DEU). There are four possibilities for the next failure: (1) If the dedicated KB fails then the common KB may serve the remaining DEUs. We have a simplex keyboard and a duplex DEU. (2) If the common keyboard fails, then DEU B has no access to a KB. We complete the mission in simplex. (3) If DEU C fails, the dedicated KB has no DEU to serve or (4) If DEU B fails, then we complete the mission with duplex KBs and simplex DEU. We need go no farther than the duplex and simplex states because our general model applies in this case. The resulting equation formulation and solution are quite complex. There

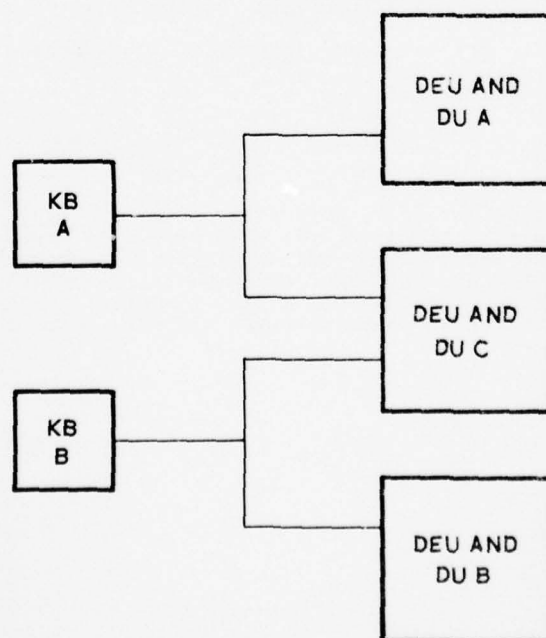


FIGURE 13 SYMBOLIC INTERCONNECTION DIAGRAM OF THE MCDS

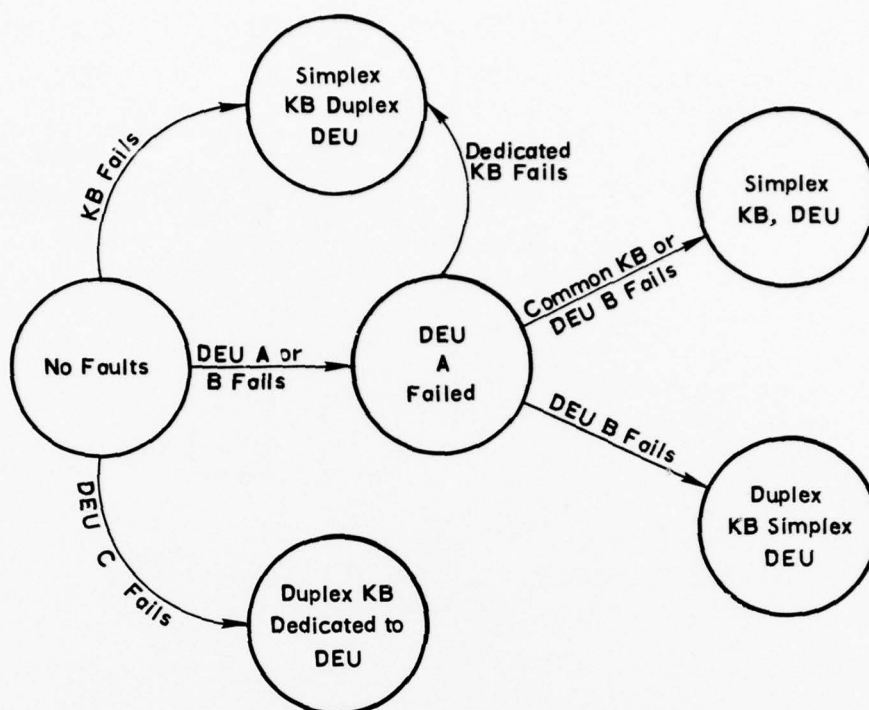


FIGURE 14 FAULT OCCURRENCE/RECOVERY STATUS STATE DIAGRAM FOR THE MCDS

are three branches from the no-faults state, so that there are four mutually exclusive events to be considered. The resulting solution covers a letter-sized typewritten page.

#### Modeling I/O Controllers Serving Several Devices

The Shuttle data processing system presented several instances in the flight-critical bus area where one controller serves more than one device. As an illustrative example, consider triply redundant controllers each serving two devices as in Figure 15. The purpose of this type of configuration is to have three copies each of devices types A and B available to the computer set wherein no two failures disable access to the devices. Faults have a different impact with this arrangement depending on where they occur. A fault in  $C_1$  disables both  $A_1$  and  $B_1$ , but a fault in  $A_1$  does not disable  $B_1$ . The device types are independent in pairs, but are in actuality dependent through the controllers. The modeling technique used in the previous sections results in mathematically intractable formulations when applied to this situation. However, a less accurate model may be formulated that was checked by the simulator. There are two extreme approximations possible with the previous modeling technique. One approach involves assuming complete unit independence and the other is to assume total unit dependence. These represent an upper and a lower bound, respectively, to the true survivability. An intermediate solution that provides realistic, usable results may be obtained by taking each of the mutually exclusive cases of the controller failure combinations and modeling survivability of the remaining devices given that failure combination. Each possible combination that can result in a successful mission is modeled. As an example of one of these combinations, suppose  $C_1$  fails and  $C_2$  and  $C_3$  don't fail. Then devices A and B must survive in duplex. To formulate the equations for the example, we recognize that there are three controller-failed conditions which allow the system to survive: none, one, or two failures. There are three ways to have one or two failures and one to have none. The survivability then becomes

$$\begin{aligned}
 S(T) = & R_C^3(T) S_3(T,A) S_3(T,B) \\
 & + 3cR_C^2(T) (1-R_C(T)) S_2(T,A) S_2(T,B) \\
 & + 3c^2R_C(T) (1-R_C(T))^2 S_1(T,A) S_1(T,B)
 \end{aligned}$$

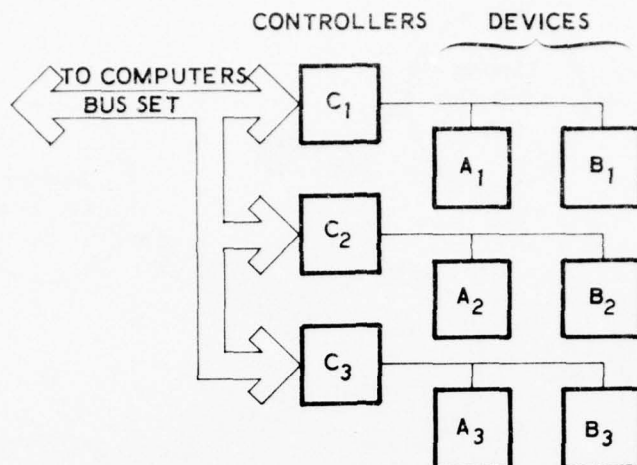


FIGURE 15 DEPENDENT I/O DEVICE EXAMPLE CONFIGURATION

## 6. SIMULATION

### Simulator Design Objectives

Much attention has been given to improving the mission success probability (MSP) of computer systems by the addition of protective redundancy. Such redundancy allows the system to continue correct operation in the presence of one or more failed components. The efficacy of this improvement is measured by the MSP increase. The mission success probability is defined as the probability that, given that there were no failed components or erroneous memory information present at mission inception, the hardware and software are operating correctly at the end of the mission. Thus the system must be able to survive both permanent and transient faults.

In order to make an accurate analytic determination of the MSP of this type of system, all fault-tolerance processes (e.g., detection, recoveries, etc.) must be modeled. However, for even a reasonable approximation to a real-world implementation, a mathematical model soon becomes intractable. Simulation is then the alternative solution. The use of simulation studies to investigate the behavior of computer hardware/software systems is well-established. Simulation is used for those situations which are intractable to an analytic approach, or for which the essence is lost when the prerequisite abstractions and simplifying assumptions necessary to the analytic technique are made.

The goal in the RCS work was an approach that is applicable to a wide variety of computer designs, and one which reflects the hardware/software interaction. Thus, a logic-level simulation would provide needless detail, in addition to sacrificing versatility. Hence, a modeling level of detail was chosen that permits description of system details, but is versatile enough to accommodate different computers and configurations. Translating these ideas into RCS simulation objectives yielded the following three items. The simulator should produce: (1) the fault tolerance of each of a wide variety of reconfigurable computer system configurations; (2) configuration performance parameters for use in analytic modeling; and (3) the behavior of a configuration in various fault environments. The requirements imposed on the simulator design by these three objectives are examined in the following paragraphs.

The simulator should be able to produce the desired measures of fault tolerance for a wide variety of configurations. This requirement can be satisfied in a reasonable way by structuring the simulator such that the various fault-detection and recovery algorithms are implemented as subroutines. Thus a configuration can be described by specifying the applicable set of subroutines, plus the necessary parameters. This simulator structure provides versatility and modularity, and minimizes the impact of addition of new subroutines.

Configuration performance parameters are those required when using the analytic model for analysis of a configuration. For example, the transient leakage in triplex,  $l_3$ , has been defined as the conditional probability that transient recovery fails, given that a transient has occurred. If a configuration is analyzed by mathematical modeling,  $l_3$  is one of the input parameters of the model. However, it is difficult for the designer to evaluate  $l_3$ , since it may depend on: the location of the transient fault; their occurrence rate  $\tau$ ; the time between occurrence and detection of a fault; and the recovery algorithm used. By introducing these factors into the simulation and gathering statistics describing the computer system reaction to transient faults,  $C_T$  can be estimated by computing the

ratio of the number of unsuccessful recoveries from transient faults to the total number of transients. Thus, for the configurations where the mathematical modeling is applicable, one simulation run gives an estimate of these parameters of the modeling. Then using the model, the MSP of the configuration can be easily determined for any given time,  $t$ .

The fault environment provided in the simulator should be sufficiently versatile to provide all expected possibilities to test the recovery algorithm utilized in the configuration under simulation. Thus low or high failure rates, existence and duration of transient bursts, long transients, mathematical fault-distribution functions, etc., must be provided. Implementation of this fault environment should be accomplished so as to provide maximum flexibility of environment choice by the user.

#### Simulator Overview

The simulator program consists of an integrated collection of FORTRAN IV computer programs organized to simulate the detection of faults in a reconfigurable computer system and the computer system's successful/unsuccessful recovery actions taken in response to the detected faults. A simulation run consists of several phases. First, the system is initialized by obtaining the input parameters and initializing fault counters. Next the system simulation begins. Faults are randomly generated for several missions and placed in a table. The fault table is searched to determine the next mission in which a fault occurs. After the mission parameters are initialized, the handling of faults is simulated. Then the statistics for the mission (i.e., final state, number of faults, causes of failures, etc.) are gathered. This process is repeated until all missions are simulated, and then estimates for analytic model parameters are calculated and printed along with the simulator statistics. A summary of the simulator utilization is shown in Figure 16.

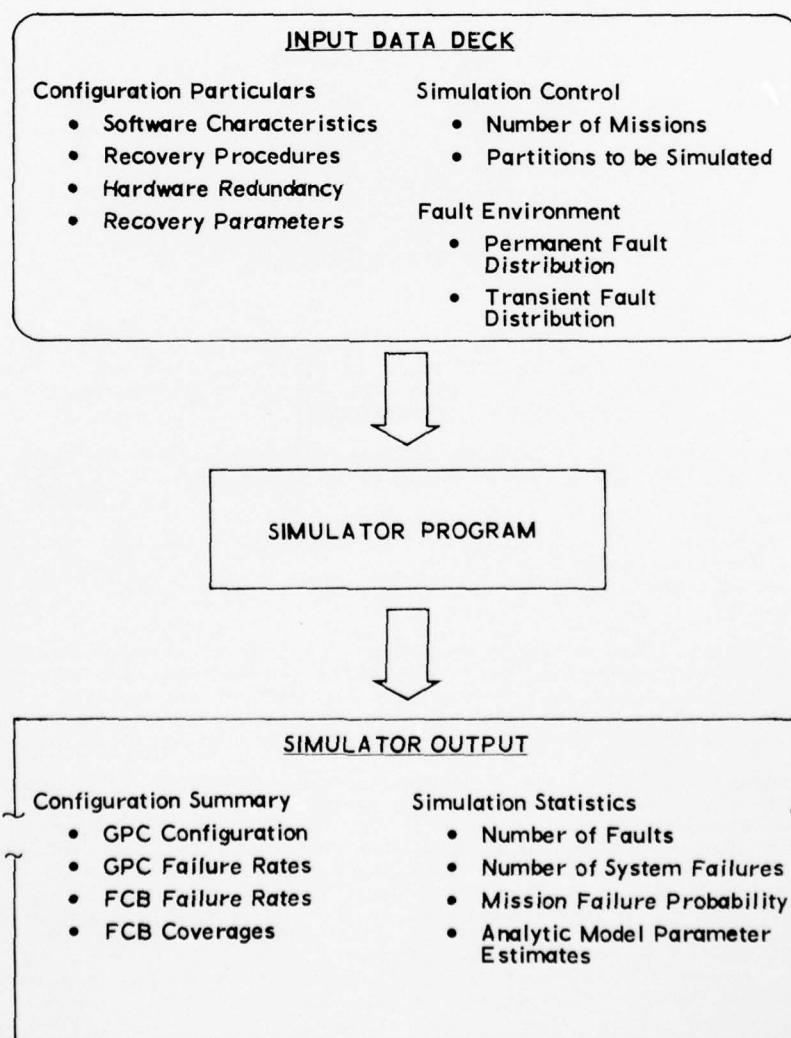


FIGURE 16 SIMULATOR UTILIZATION SUMMARY



A variety of configurations can be specified by the input parameters. These configurations differ in: (1) the degree of hardware redundancy and the way this redundancy is utilized; (2) the methods employed for detection and isolation of faults and their effectiveness; (3) the recovery procedures that are used and their performance characteristics; and (4) the software scheduling mechanism and parameters. The simulated computer system may contain up to five computers configured as an adaptive quintuplex system that can reconfigure only to a duplex system, or all the way down to a simplex system. The effects of a variety of reliability enhancement techniques can be incorporated into the configuration by adjusting the values of several input parameters. For example, the use of an NDRO memory is taken into account by adjusting the program integrity parameter, and memory parity is accounted for with the memory BITE effectiveness parameter. The simulator accounts for system recovery from transient faults as well as permanent faults. Currently the roll-ahead, rollback, memory copy, and "wait and see what happens" transient recovery techniques can be specified for a configuration. Transient recovery is specified in the input as (1) the recovery techniques used, (2) the recovery duration, (3) the effectiveness of the recovery technique for each class of fault, and (4) the recovery recurrence interval. The simulator also takes into account global software characteristics such as: the type of scheduling mechanism, the minor and major cycle durations, and the relative minor-cycle-program size.

The fault environment is defined in terms of probability distribution functions for random variables representing the fault-occurrence time, the fault duration, and the fault location. Permanent faults are assumed to have interarrival times which are exponentially distributed, and infinite fault durations. Here it is assumed that permanent faults occur independently of each other and are memoryless (the probability that a fault occurs between the times,  $T$ , and  $T+\Delta T$ , is independent of  $T$ ). Transient-fault interarrival times may be either exponentially distributed or burst-distributed. The burst distribution assumes that transient faults have a tendency to arrive in groups. The random variable representing the transient-fault duration may be uniformly distributed or exponentially distributed; other distributions for fault occurrence times and durations can be defined by the simulator user with only minor modifications.

After all missions have been simulated, the accumulated system statistics are displayed and used to estimate system performance measures. The mission failure probability is estimated by finding the ratio of the number of system failures to the number of missions simulated. Global parameters required by the analytic model are estimated from the ratios of other simulation statistics. For example,  $C_2$  which is the probability that the system recovers given that a fault occurs, is required by the analytic model. It is estimated by the ratio of the number of successful transitions from duplex to simplex to the total number of attempted recoveries in duplex. The reliability of a parameter estimate is dependent upon the number of samples used. The simulator calculates this confidence interval and prints it out along with the estimated parameter.

#### Simulator Model Formulation

The approach taken in the formulation of the simulator is an extension of the approach described in Ref. 4. Formulating the simulator in this way permits the computer system to be viewed as a finite state automaton. Thus, the system is described by the states it may assume and the possible transitions between states.

The computer system states are defined by two conditions. The first of these is the function being performed by the system. Examples of these are:

- a. Normal Operation;
- b. Recovery Operation;
- c. Reduced Capability Operation;
- d. System Restart; and
- e. System Failure.

The second of the system-state defining conditions is that of the number of permanent faults that the simulated system has suffered during the particular simulated mission under consideration. Obviously, the system that has not yet encountered a fault will be in normal operation, while a system that has encountered faults may be in recovery operations, reduced capability operations, system restart, or may have failed.

Transitions between states in the simulated computer system will be caused by either of two events. The first event that may cause a transition is the detection of a fault. For example, the first detection of a fault in the Shuttle GPC set causes a transition to the delay-reconfigurable state which simulates the FCOS transient-recovery method. Later detections of faults will cause a state transition in the simulated system. The second event, the completion of a recovery procedure, will definitely cause a transition to another state. What state is the destination of this transition depends on the type of recovery procedure attempted. For example, the successful completion of a normal recovery procedure when four GPCs are operating will return the simulator to the transient operations state. However, a recovery procedure that requires deactivation of one of three GPCs will cause the simulated system to degrade to the duplex state.

An important aspect to be noted when considering the organization of the RCS simulator is that it is an "event-driven" simulation. Thus, the initial state transition is only made when an event, in this case either a permanent or transient fault, occurs. Use of this type of structure provides a significant saving in computer time.

Figure 17 presents the simplified state diagram of an adaptive NMR configuration that employs rollahead, rollback, and memory copy for transient-fault recovery. A count of the number of active computers is maintained so that states I, II, III, and VII do not need to be duplicated for the operation of more than three computers. In the Normal Operation state with three or more computer units, the outputs of the computers are periodically compared. Disagreement of one or more computers constitutes fault detection and requires exit from this state. As long as two computers are fault-free, the rollahead recovery procedure is used and, if it is not successful, the memory copy. If all computers disagree at the same time, a system restart is initiated.

The Rollahead state is entered to simulate the computer system's attempt to recover from a detected single fault. The state vector (consisting of program variables and all register contents) of one good computer is used to replace the non-agreeing computer's state vector. However, all transient failures are not corrected by this procedure since a bad instruction cannot be restored. The approach taken in the simulation is to provide for the specification of a rollahead success probability. This probability can be formally defined as:

$$P_{\text{suc}} = \text{Pr} [\text{Fault is corrected given that a fault has occurred, has been detected, and its physical cause has disappeared when correction begins}]$$

An analysis, which gives consideration to the type of memory (e.g., 2 1/2D, 3D, DRO, NDRO, etc.) and the consequences of memory faults, will yield an estimate of the rollahead success probability (or program integrity).

The Memory Copy recovery procedure is entered after a specified number of rollaheads have been completed unsuccessfully. The memory contents of one good memory are transferred into the faulty memory. In order to avoid interruption of computation, the transfer is effected on the basis of cycle stealing. It ends with the updating of the state vector of the faulty computer. Since, during a memory copy, normal application routines continue, it is possible that a new fault shows up. The following (conservative) assumption has been made in order to simplify the simulation. Upon detection of a second fault during a memory copy, the memory copy procedure is abandoned and the computer for which this memory copy was intended is discarded. It is assumed that memory copy provides recovery from transient faults which have disappeared when the memory copy began with a probability equal to the memory copy efficacy.

The System Restart state is entered when all computers disagree upon comparison. The recovery procedure from this state may consist of a memory verification. Relevant memory locations are read, voted upon, and restored. Extensive diagnosis may also be run. Finally, if a backup memory is available, reloading may be possible. Then the application program is reinitiated from the restart point. After a successful system restart, the system returns to the normal operation state. However, since all computers stop their normal computation during a system restart, this recovery procedure is time-critical. Note that in a benign fault environment, the probability of having a system restart is quite small ( $\approx 1$  for 1 million faults). However, system restart is necessary if the fault environment is so harsh that bursts of faults can hit several computers at a time or if the probability of a short power failure is not negligible.

If a spare is available, it should be activated once a permanent fault has been recognized. As part of the activation process, the spare is checked and conditioned by one of the good computers. In the situation depicted in the state diagram of Figure 17, spares are not available for the duplex and simplex simulation. This is thought to be compatible with the expected applications.

The Duplex operation state is entered upon the determination that a permanent fault exists in one of the three computers in the computer system. This state is quite similar to the normal operation (N units) state, except that the only available recovery procedure is program rollback.

The Rollback state is entered upon the detection of a fault when the computer system is in the Duplex operation state. Rollback is the term used to describe repetition of the program segment executed just prior to the detected output disagreement. The state vector at the beginning of each program segment is maintained in order that the rollback procedure may be accomplished. After the program segment has been repeated, the outputs of the two computers are compared; if the correction is successful, the computer system switches back to the Duplex operation state. If the output differs, the system rolls back again; this unsuccessful recovery process continues a predetermined number of times before changing the computer system state to diagnosis. Since both of the active computers remaining in the computer system must stop their normal computations during a rollback, this computer recovery procedure may be time-critical. However, if comparisons are frequent enough, a rollback should not last more than a few milliseconds.

A disagreement upon comparison in duplex does not indicate which of the computers produced the wrong value. Thus the Diagnosis state must be entered. To accomplish diagnosis, self-tests are run. If they are successful, the faulty computer is isolated and the system switches to simplex. If unsuccessful, the system is unable to decide which computer is faulty and the system fails. Diagnosis programs are obviously time-critical. Note that it would be possible to include a memory copy which would take place

once a diagnosis had been successful: the memory of the good computer would be copied into the bad one. However, this improvement is not as good as it would seem since many transients cannot be detected through diagnosis.

In Simplex operation, comparison is no longer available for detection of faults. We must rely mostly on the BITE to detect faults. CPU transients are difficult to detect. Some may be caught through go/no-go counters and storage protection. Memory faults are easier to detect. Parity check is especially useful. When a fault is detected, a rollback is initiated. If the fault is not detected, a failure occurs.

Rollback in Simplex is the same procedure used in duplex. Since it is the only recovery algorithm available in simplex, it is repeated as long as it is not successful. If recovery from the fault cannot be effected, a system failure will occur when the system has been down too long.

The System Failure state is entered when the system is unable to run properly and longer or when computational requirements have not been met for too long a period of time. Upon recognition of the condition of a system failure, the Driver program discontinues the simulation of a mission.

Causes of failures are:

- a. Excessive time in rollahead, memory copy, or rollback: This should not happen since the system must be designed so that a recovery procedure does not endanger it. However, it might happen that the continuous repetition of such procedures could be fatal for the successful completion of the mission.
- b. An overly long system restart: A system restart is a very rarely called procedure. But it is long (a few seconds), and may not always be tolerable.
- c. Diagnosis incomplete when available recovery time expires: Normally, diagnosis follows rollback. It is possible that these two recovery procedures sometimes take too long.
- d. Undetected faults in simplex.
- e. A too long rollback in simplex: This happens when a permanent occurs or when a non-recoverable transient occurs.
- f. IOP failures: In the case of non-dedicated EEMs, the system fails when all IOPs fail or when all but one fail and the computers are unable to decide which is the good EEM.
- g. Bus failures: The system fails when all buses fail or when all but one fail and the computers are unable to decide which is the good bus.
- h. Actuator/sensor failures.

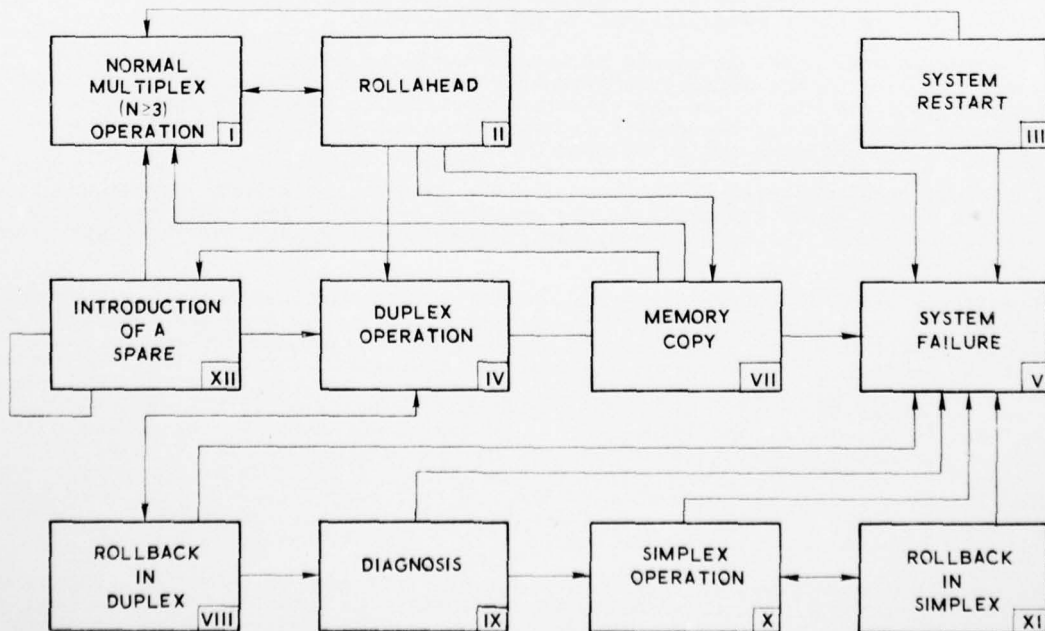


FIGURE 17 SIMULATOR STATE DIAGRAM



## Simulator Inputs

The inputs required by the simulator are summarized in Table I. The use of some of these inputs is discussed below. The detection probabilities are the probabilities that a computer detects its own faults (except through diagnosis). This is not significant for NMR configurations ( $N = 3$ ) since all faults are detected and located through voting or comparison. However, these probabilities become critical in duplex and simplex. In duplex, faults are detected through comparisons. However, BITE or self-test is needed to isolate the faulty computer. In simplex, BITE is necessary, since it provides the only means for detecting transient faults. For simplex operation the detection probability of CPU faults is low. Faults in the CPU usually cause only a wrong output which will not be detected by BITE. However, some will be detected. Those are the ones which cause a forbidden address to be computed or those which modify the computing sequence in such a manner that a go/no-go counter detects them. IBM estimates this detection probability to be about 35%. The main technique to detect a memory fault is parity encoding. When it exists, the probability of detecting a memory fault is usually better than 80%. When it does not exist, this probability is quite small. Self-test programs (diagnosis) are run in a duplex system where a fault has been detected but not isolated. Note that if the fault is transient, the self-test will probably not diagnose it, since it usually dissipates before the test is run.

If the configuration includes some additional hardware for the Input/Output Processor, the consequence of faults in this hardware has to be assessed. We partitioned the configurations in two classes. In the first class (dedicated IOPs), we assume that a fault in the IOP is equivalent to a fault in the computer and sometimes on the corresponding bus. In the second one (non-dedicated IOPs), we assume that IOPs are independent from the computers. The system can work as long as one computer and one IOP are good. Note that the dedicated case includes software TMR.

In the present simulator, the recovery procedure for an NMR system is the state vector transfer. Memory copy is optional. Once a recovery procedure has failed for a certain fault, it is useless to attempt to recover through the same procedure. Some other one has to be chosen. If after completion of a recovery procedure a fault recurs in the same computer after a time less than the unacceptable recurrence interval, the system decides that the recovery procedure was unsuccessful and attempts something else. Usually, the

**TABLE I REQUIRED SIMULATOR INPUTS - GPC PARTITION**

NUMBER OF SIMULATED MISSIONS
MISSION-DEPENDENT PARAMETER
Mission Time
MACHINE-DEPENDENT PARAMETERS
Permanent Failure Rates
BITE Detection Probability of a CPU Fault
BITE Detection Probability of a Memory Fault
Self-Test Program Efficiency
Self-Test Program Duration
CONFIGURATION-DEPENDENT PARAMETERS
Number of Computers
Number of Spares
Dedicated/Non-Dedicated IOPs (Input/Output Processor)
Probability that an IOP Fault Hits the Bus
Number of Non-Dedicated IOPs
Applicable Recovery Algorithms
Recovery Algorithm Characteristics
Duration
Unacceptable Recurrence Interval
Maximum Number of Rollbacks
Program Integrity
Memory Copy Efficacy
SCHEDULING PARAMETERS
Iteration Period
Time Between Comparisons
Major and Minor Cycle Durations
Asynchronous/Synchronous Mechanism
ENVIRONMENT-DEPENDENT PARAMETERS
Transient Failure Rates
Transient Failure Duration



recurrence intervals will be chosen equal to the duration of one major cycle. The rationale is that the memory is thoroughly exercised in one major cycle. The memory-copy efficacy is the probability that a memory copy corrects a transient fault. The only reason why it would not succeed is that the transient might have hit the microprogram initiating the memory copy. This is very unlikely since this program resides in a read-only-memory or microstore. The roll-ahead efficacy and the rollback efficacy for faults that do not cause program memory damage must also be specified, but they are generally assumed to be one. The program integrity is listed with the other recovery algorithm characteristics because a transient recovery algorithm not involving memory refresh cannot succeed when there is a program memory damage. Program integrity is strongly linked to the type of memory: an NDRO memory is much better in this respect than a DRO memory. The fact that there is no need to restore the information makes it very unlikely that a transient fault damages instructions or constants. In addition, in most NDRO applications, the write voltage for the program memory is disabled except when altering the program under AGE control.

An important point in the application of CAST to the Shuttle data processing subsystem is the determination of simulator input parameters. There are several methods for obtaining them if their values are not obvious: Failure rates and built-in test detection probabilities are usually obtained from the manufacturer. Parameters affecting transient fault recovery such as the Program Integrity or transient leakages can be determined by engineering analysis or by logic level simulation. Parameters that couldn't be obtained from the manufacturers were estimated by an engineering analysis. One of the required simulator inputs is called program integrity (PI). This simulator input is the probability that a transient fault in the GPC memory does not alter a program word.

We use a "top-down" approach by subdividing the GPC memory into functional components and then in turn further partitioning these functional components. For each transient failure mode within a component we determine whether memory will: always be corrupted; be corrupted only if the component is used; or never be corrupted. The expression for the program integrity can be written as one minus the probability that a transient fault alters a program word. Thus PI is written as

$$PI = 1 - \left( \sum_i n_i \sum_j \beta_{ij} \tau_{ij} \right) / \left( \sum_i n_i \sum_j \tau_{ij} \right)$$

where:  $\tau_{ij}$  is the rate of occurrence of transient failure mode  $j$  in component  $i$ ,

$\beta_{ij}$  is the probability that transient failure mode  $j$  in component  $i$  corrupts memory, and

$n_i$  is the number of components of type  $i$ .

The first partitioning of a 16K - 2 1/2D core memory as found in the IBM-4 AP-101 basic configuration is shown in Figure 18. This partitioning divides the memory into the timing page and four storage pages.

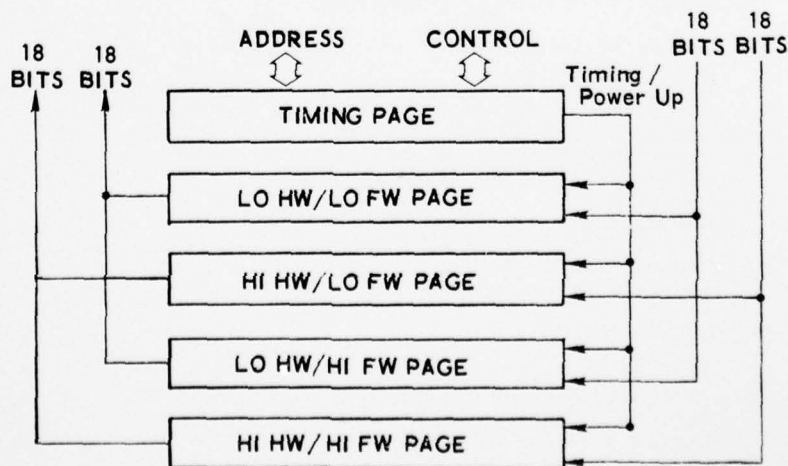


FIGURE 18 PARTITIONING THE AP-101 MEMORY

Further partitioning continues as shown in Figure 19 for a storage page. We see from this partitioning that a transient in the output buffer will only corrupt the memory output, but a transient in the data register would surely corrupt memory during the restore cycle as well as the memory output.

Consider the case of a Y-driver as shown in Figure 20. If a transient strikes a powered Y-driver, then any Y-driver failure mode will corrupt memory during the read and/or restore cycle. The quantity  $\beta_{ij}$  for a Y-driver then becomes the probability that it is selected while a transient is active. The Y-driver on the page has a 1/32 probability of being used, and for a 16K memory, the page of the driver of interest has a 50 percent probability of being used. If we assume program words are accessed every 3  $\mu s$ , then the quantity  $\beta_{ij}$  for one Y-driver becomes

$$\beta_{ij} = 1 = \sum_{n=1}^{\infty} \left(\frac{63}{64}\right)^n P(T_d = 3n \mu s)$$

where  $T_d$  is a discrete random variable representing transient duration. If we assume it is uniform from 3  $\mu s$  to 300  $\mu s$  at intervals of 3  $\mu s$  for ease of computation, then  $\beta_{ij}$  becomes

$$\beta_{ij} = 1 - \frac{1}{100} \sum_{n=1}^{100} \left(\frac{63}{64}\right)^n = .57$$

Computing the  $\beta$ 's as above for the remaining functional components and finding the  $\beta_{ij}$ 's as is done for permanent faults, program integrity is found to be .30.

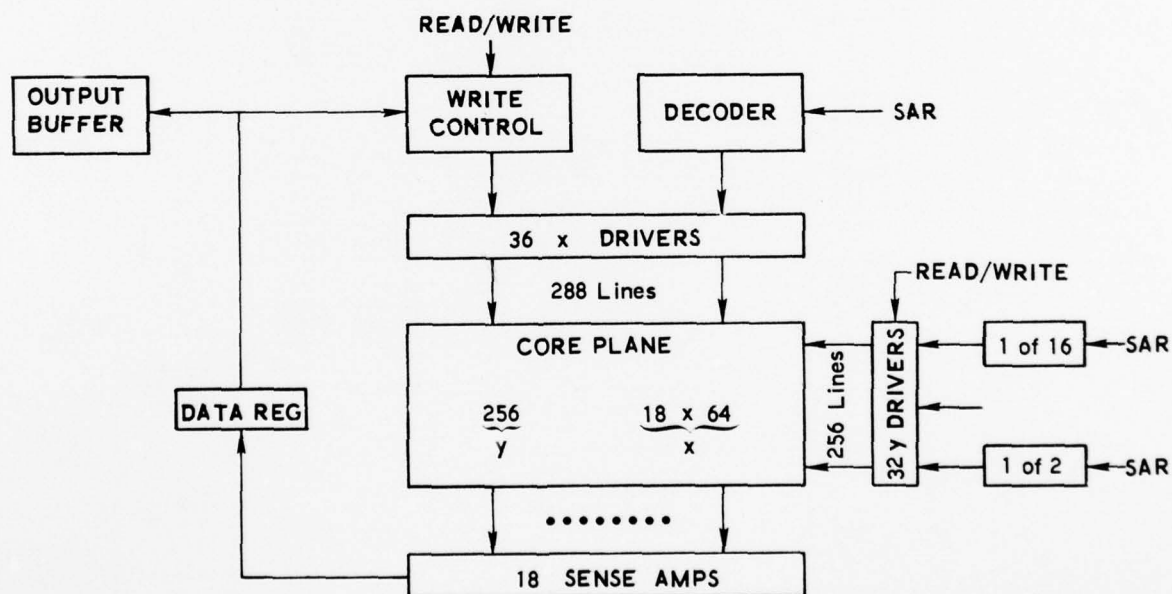


FIGURE 19 PARTITIONING OF THE AP-101 MEMORY STORAGE-PAGE

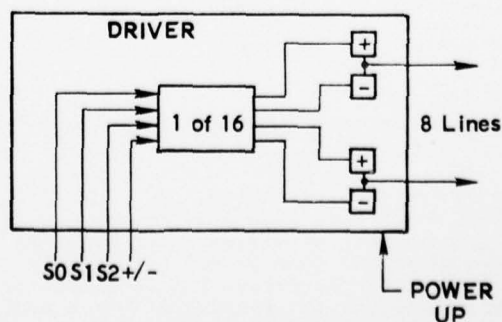


FIGURE 20 Y-DRIVER ANALYSIS

### Input/Output Subsystem Simulation

In general, the input/output network is not specified by the simulator's input deck because the configuration of the I/O devices associated with a data processing system is very application-dependent. For a given application, a set of routines must be programmed to simulate the system's handling of faults occurring in the I/O equipment group. These routines are invoked upon occurrence of an I/O fault to simulate the entire fault-recovery process (i.e., fault detection, faulty device isolation, and system recovery) and return information concerning the impact of the fault on the system to the calling program. After they are programmed, the I/O simulation routines must be interfaced with the RCS simulator program. Simulation routines were developed for the flight-critical bus subsystem of the ALT space Shuttle and the methodology employed could be used for other I/O subsystems.

Figure 21 shows the layout of the flight-critical bus (FCB) equipment group of the space Shuttle (ALT). The eight flight-critical buses, FC1 ~ FC8, are interfaced with all GPCs (computers). Each dedicated display unit (DDU) is interfaced with three buses by means of three redundant ports. Each flight-forward Multiplexer-Demultiplexer (MDM) is interfaced with two buses by means of a primary port and a secondary port. If the electronics associated with a primary port fails, the backup port is switched in. Each interface unit (MDM or DDU) controls several dedicated and/or non-dedicated devices (non-dedicated devices are shaded and can be accessed through more than one MDM). These devices are redundant (e.g., ACCEL1, ACCEL2, and ACCEL3 perform identical functions), thus one of them can fail without causing a system failure.

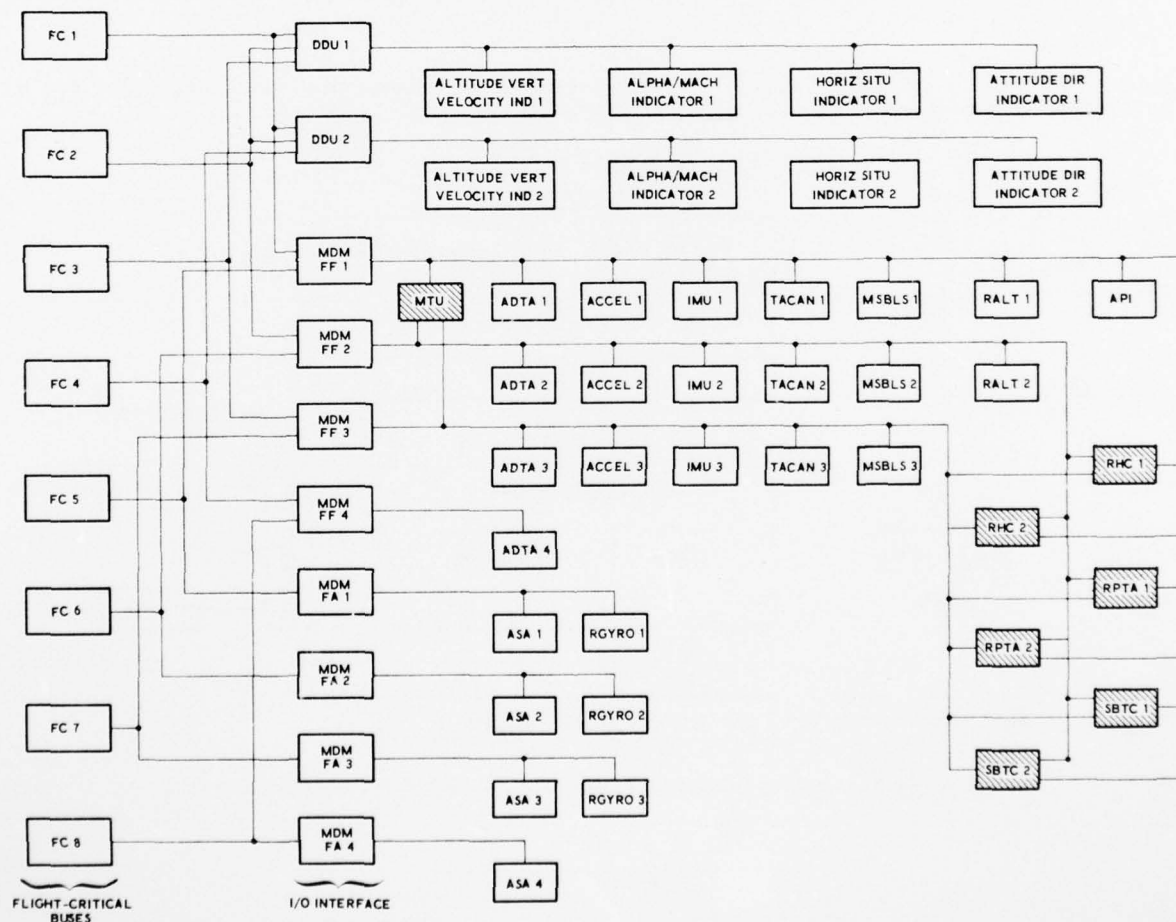


FIGURE 21 FLIGHT-CRITICAL BUS CONNECTIONS (ALT)

It was impractical to use the same method for modeling the flight-critical bus equipment group as was used to model the computers because the FCB equipment group's complexity would have resulted in a multitude of states. The behavior of the I/O equipment group is represented by a set of tables and some procedures. The tables define the current state of the system, i.e., the device redundancy, the device interconnections and the device status. The procedures define the fault-induced system action, the resulting table modifications (i.e., state transition) and the successfulness of recovery. Both the built-in test equipment and the redundancy management software are factored into the implementation

of these procedures, since they determine the fault detection, isolation and recovery success probabilities.

For example, the interface between the flight-critical buses and the Interface Units (IU) is reflected by Figure 22. Each row corresponds to a flight-critical bus and each column corresponds to an IU. An element that is indexed by a particular bus and IU (row and column) is assigned to a number according to the following scheme:

- 0 - The bus does not have a functional interface with the IU.
- 1 - The bus has an active interface with the IU.
- 2 - The bus has a functional, but inactive, interface with the IU (i.e., this represents a secondary port).

Thus from Figure 22, it can be inferred that MDM FF1 is interfaced with flight-critical buses FC1 and FC5. FC1 is connected to the primary (active) port of MDM FF1, and FC5 is connected to the secondary port. Note that each DDU has three active ports. Here it is assumed that display information is transmitted on buses FC1 - FC4, and the actual bus used by a DDU is selected by a manual switch on its control panel. The interface between the interface units and their associated devices is represented by **additional tables**.

	D	D								
	D	D	F	F	F	F	F	F	F	F
	U	U	F	F	F	F	A	A	A	A
	1	2	1	2	3	4	1	2	3	4
FC1	1	1	1	0	0	0	0	0	0	0
FC2	1	1	0	1	0	0	0	0	0	0
FC3	1	0	0	0	1	0	0	0	0	0
FC4	0	1	0	0	0	1	0	0	0	0
FC5	0	0	2	0	0	0	1	0	0	0
FC6	0	0	0	2	0	0	0	1	0	0
FC7	0	0	0	0	2	0	0	0	1	0
FC8	0	0	0	0	0	2	0	0	0	1

FIGURE 22 BUS - IU INTERCONNECTION MATRIX

#### REFERENCES

1. E. Parzen, Modern Probability Theory and Its Applications, New York, Wiley & Sons, 1960, pp. 251-263.
2. W. B. Davenport and W. L. Root, Introduction to Random Signals and Noise, New York, McGraw-Hill, 1958.
3. W. G. Bouricius, et al., "Reliability Modeling for Fault-Tolerant Computers," IEEE Trans. Comput., Vol. C-20, No. 11, November 1971.
4. J. Kruus, Coordinated Science Laboratory, University of Illinois, "Upper Bounds for the Mean Life of Self-Repairing Systems," 1963, Report R-172.

#### ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions of Mr. H. Olivier Lévy who designed and implemented the first version of the simulator. His work was a major factor to the success of the first phase of the work reported here. In addition, the authors wish to recognize the contributions, through stimulating discussions and suggestions, of Dr. A. A. Avižienis, Mr. S. R. Pond, and Dr. D. A. Rennels. Finally, the authors wish to acknowledge the patience and contributions of the NASA sponsors, Messrs. B. L. Dove and J. L. Spencer of Langley Research Center, and Messrs. A. E. Brandli and C. D. Warnick of Johnson Space Center.



AD-A041 042

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT--ETC F/G 1/3  
INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS.(U)  
1977

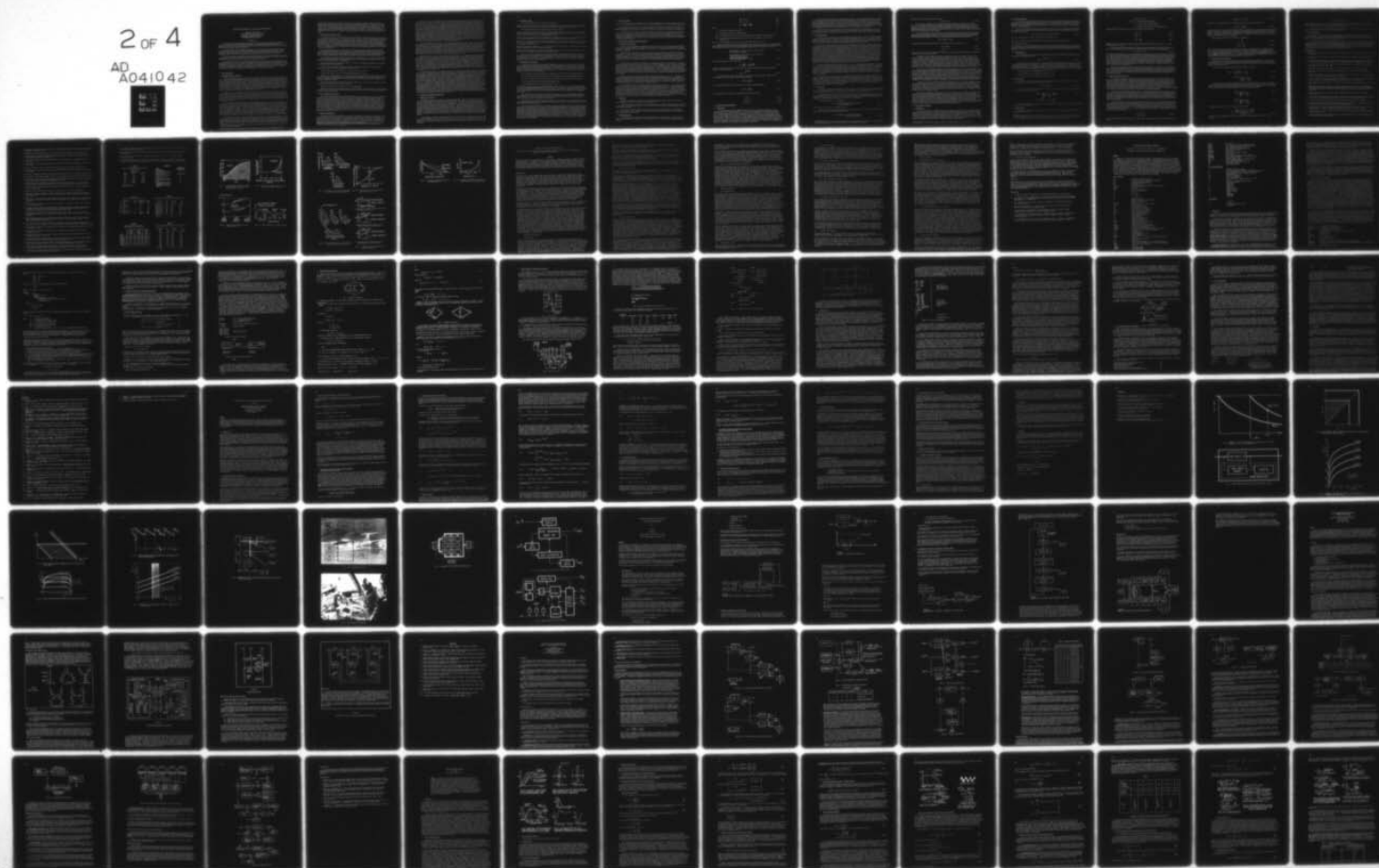
UNCLASSIFIED

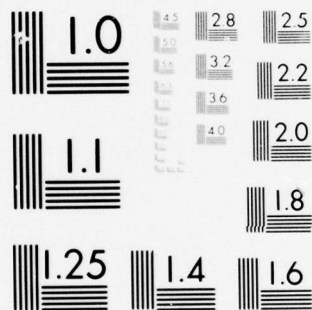
AGARD-06GRAPH-224

NL

2 OF 4

AD  
A041042





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

## SOFTWARE RELIABILITY: ANALYSIS AND PREDICTION

Martin L. Shooman\*  
 Department of Electrical Engineering  
 Division of Computer Science  
 Polytechnic Institute of New York  
 Brooklyn, New York 11201

## SUMMARY

With the advent of large sophisticated hardware-software systems developed in the 1960's, the problem of computer system reliability has emerged. The reliability of computer hardware can be modeled in much the same way as other devices using conventional reliability theory; however, computer software errors require a different approach.

The paper begins by describing the types and causes of software errors and provides working definitions of software errors and software reliability. Some of the basic data on frequency of occurrence of errors is then discussed. The paper then summarizes and references some of the software reliability models which have been proposed and concentrates on one developed by the author.

This newly developed probabilistic model predicts reliability based on the initial number of errors in a program, the number removed, and the number remaining in the program. The model constants are calculated from operational test data taken on the software performance.

The calculations result in a decreasing probability of no software errors versus operating time (reliability function). The rate at which the reliability decreases is a function of the man-months of debugging time. Similarly, the mean time to occurrence of operational software errors (MTTF) is obtained. The MTTF increases slowly and then more rapidly as the debugging effort (man-months) increases. The model permits estimation of software reliability before any code is written and allows later updating to improve the accuracy of the parameters when integration or operational tests begin.

## 1.0 INTRODUCTION

## 1.1 The Age of Large Computers

The first question that one hears when the term software reliability is mentioned in discussion is, what is that? As the digital computer continues to pervade more and more of our modern technology, we rely on its output more and more for control, data recording, analysis, and decision making. Thus, the size and complexity of the required tasks, the computer hardware, and the computer software has drastically increased in the last three decades. With such huge size and complexity, it is virtually impossible to definitively specify the problem (without error), to make failure free computer hardware, and to remove all errors from the software. The engineer must pose and help to answer the following problems: How often can the system fail in use and still be considered acceptable? What percentage of these failures is miswritten or misinterpreted specifications? What percentage is hardware failures? Lastly, the subject of this paper, what percentage is due to software errors?

During the decade of the 1940s computers were born. In the 1950s commercial hardware and assembly programs became available. In the 1960s programming languages became popular, complex operating systems emerged, hardware became huge and sophisticated, and large programs consisting of several hundred thousand words of code became the norm. By the 1970s one began to speak of automatic programming, huge virtual memory and interconnecting networks of computers. Along with these new developments one begins to speak of programs containing millions of words of code and performing huge and complex real-time tasks.

The growth in computer hardware can be illustrated in several ways. First of all the total number of computer installations in the United States has grown phenomenally (see Table 1-1) reaching about 50,000 in 1970. Sackman<sup>1</sup> quotes an estimate that the European Economic Community will have approximately 10,000 computers by 1970. Most of the above mentioned computers are large general purpose scientific and business computers. There are also many special purpose military and industrial computers. A detailed count is difficult to obtain; however, in Ref. 3, an estimate of the current U.S. Air Force annual expenditures for computer hardware is quoted between \$300 and \$400 million per year. The corresponding estimate of the cost of procuring computer software is \$1,000 million to \$1,500 million!! Future estimates of the hardware vs. software split in costs is shown in Figure 1-1. Thus, software costs already exceed hardware costs by a factor of 3 or 4 in the U.S. Air Force and are rising rapidly.\*\*

Along with this growth has come a realization that the largest effort in developing software is due to software integration, test, correction, retest, operational release, correction and rerelease. Actual writing of the first set of code is a small task in comparison. An even more compelling observation is that computers are increasingly being used as the heart of complex real-time systems such as air traffic control, vehicle control, space systems, and military systems. System reliability is the most important

\* This work was supported by the Office of Naval Research, Statistics and Probability Program under contract N0014-67-A-0438-0013.

\*\*Recent estimates on the cost of software to the entire U.S. economy range from \$10-\$19 billion.

performance measure in such systems. After all, if a large batch computer crashes occasionally, it probably only means the users enjoy one rather than three turnarounds that day.\* Similarly, if a time-sharing system goes down infrequently for half an hour, 50 annoyed users have an unanticipated coffee break.\* It is a far different story if the air traffic control system handling the metropolitan New York area crashes on a stormy night under saturated conditions, and requires a several-minute system re-loading delay before targets again appear on the video display terminals! In any of these situations, it is clear that a vital measure of computer system performance is both the hardware and software reliability.

## 1.2 Hardware vs. Software Errors

I am sure the reader agrees that software is a costly and time consuming product, but why must we worry about software errors? Aren't all or the majority of software errors removed during the debugging and test phases? Unfortunately not. Experienced software managers are fond of telling "war stories" (not unlike the retired general who tells everyone about his experiences in the great war) about difficult and troublesome bugs. For example, Ref. 3 cites "a software error aboard a French meteorological satellite caused it to 'emergency destruct' half of its force of weather balloons instead of interrogating them." In another case, that of the Apollo spacecraft guidance computer<sup>5</sup>, the similarity of program names led to the wrong program being called which destroyed the guidance systems parameters, necessitating a lengthy reinitialization.

The most dramatic story of a software error comes from the popular American science fiction movie, "2001 A Space Odyssey,"<sup>4</sup>. The dialogue occurs between Mission Commander David Bowman and the computer HAL while the space ship controlled by HAL is on its way to the planet Jupiter. HAL detects and Bowman replaces a faulty component. Bowman orders, "HAL, Carry out fault prediction tests." (on the removed module) "Circuit fully operational," reported HAL after only ten seconds.

But Mission Control following these actions has its own interpretation. "...this is mission control ... there is another possibility. Your computer may have made an error in predicting the fault. Both our own HAL computers agree in suggesting this..."

Bowman drummed his fingers on the console ... "HAL...is something bothering you - something which might account for this problem?"

"Look, Dave, I know you are trying to be helpful. But my information processing is normal... check my record; you will find it completely free from error."

"I know about your service record, but...anyone can make mistakes."

"I don't want to insist, Dave, but I am incapable of making an error."....

"Hello, this is Mission Control. We have completed the analysis of your AE-35 difficulty, and both our HAL computers are in agreement. The trouble lies in the prediction circuits, and we believe that it indicates a programming conflict which we can only resolve if you disconnect your HAL computer and switch to Earth mode control."

As Bowman began to switch off the computer, HAL fought back, and Bowman only gained control of his spaceship by dismantling the computer's memory.

Our software problems haven't gone this far yet, or have they?

Although the above three examples all dealt with space missions, mainly because they make the point so vividly, many similar examples of the problems caused by software failures in military and industrial computer systems can be cited.

## 1.3 Some Computer Failure Data

We may shed further light on the distribution between hardware and software errors by considering some actual data. The data given in Table 1-2 lists hardware and software failure rates for a typical real-time data acquisition system. The data represents 9 months of operation totaling 1701 hours. Inspection of the data shows that 48% of the failures were due to software. This is a startling figure if one realizes that although most computer projects estimate and predict the hardware reliability, and use redundancy techniques and high reliability parts to improve the hardware reliability, the software is left to the skill and hard work of the programming team with no quantitative assessment of design progress. We may obtain a typical estimate of what type of system reliability can be obtained in practice. Assuming a simple failure model, we may compute the mean-time to failure, MTTF as the reciprocal of the failure rate. Thus, the  $MTTF \approx 30$  hours for the data in Table 1-2. In other words about one failure per day will occur if the equipment is used 24 hours/day and one failure every three days if used 8 hours/day. (Additional data is given in Ref. 37.)

## 1.4 Computer Engineering

Historically, hardware reliability has always been a discipline which logically fell within the scope of system engineering. Similarly, software engineering must logically include software reliability. One huge difference exists in this analogy. In the late 1940s and early 1950s when hardware reliability was born as a discipline, hardware systems engineering was already a well established and exercised field. Unfortunately, software production is still largely an art so that evolution of the field of software reliability must occur in parallel with the development of software engineering.

\* Although less dramatic, such errors if too frequent can have severe economic and operational effects on batch and time-shared computation.



In this section we try to briefly discuss some of the scant knowledge of software engineering which directly relates to software reliability. One facet inherent in any engineering design is schedule and manpower needs. In Fig. 1-2 we see estimates of the speed at which code is produced measured in lines of machine code per man hour. If we approximate the median between the 10th and 90th percentile in Fig. 1-2, we obtain: in 1955 machine language programming productivity was 200 machine instructions/man month; in 1970 programming in a higher level language has raised this figure to about 700 machine instructions/man month; the projection for 1985 using structured programming techniques (top down, go to free, cf. Ref. 23) is 1000 machine instructions per man month. At the outset we might say that the switch from machine language to higher level language programming (FORTRAN, etc.) was of great help in that it increased productivity by a factor of 3.5. A deeper consideration of these facts makes this progress appear less spectacular. In the compilation process each line of code written in a higher level language expands into between 5 and 10 lines of machine code. Thus, we might say that in going from machine to higher level language we have decreased productivity by a factor of two due to the time to write an instruction, yet increased productivity by a factor of about seven due to the expansion of code by the compiler. This is only part of the story since this only measures code productivity. From Table 1-3 we see that this phase only represents from 15% to 25% of the total effort.

If we use the SAGE data in Table 1-3, coding represents 15% of the effort. Also, using the previously quoted figure that a programming project done in a higher level language proceeds at a rate of about 700 instructions per man month or 8400 machine instructions per man year, yields the following conclusions: A 12,000 word FORTRAN program would yield about a 84,000 word machine language program and would require about 10 man years to code the program. In addition, about 30 man years would be required for checkout and test (debugging). Also, about 27 man years would be required for analysis and design. One must take care in working with these gross rules of thumb to remember that time can only be traded for manpower within reason. The favorite illustration is: it takes one woman 9 months to produce a baby but 9 women can't produce a baby in one month. (see ref. 39).

Suppose two years' time were allotted for the above example. A reasonable preliminary schedule might be to start with a design group of 27 working during the first year. After 6 months, the analysis group should have enough work done so that coding can begin, so a group of 10 programmers join the project after 6 months and should be completed with work at the 18 months mark. After about 9 months a group of 10 programmers could form a test team to start testing and debugging their first sections of code which have been produced. After about 1 year, another group of 17 programmers could join and form one or more additional test teams and begin debugging on the new sections of code being written. Notice that at the peak manpower level about 64 people are working on the project!! If we assume that with overhead the salary of an analyst is \$40,000 and that of a programmer is \$30,000, the overall labor cost of such a piece of software is over \$2,000,000. (A comprehensive treatment of software cost estimating appears in Ref. 38 and 43.)

We now leave the subject of schedules and turn to some of the other measures of system performance which are used to evaluate a program: core size (number of machine language instructions), running time of the program, load factors, ease of change, and portability. When we say core size, we mean memory in general and with the advent of low cost electronic memory and inexpensive disk storage units, this is a less significant constraint than it once was. However, in space and aircraft applications where memory size is fixed by power, weight, and volume considerations, investigations have found<sup>6</sup> that if the programmers have a fixed core size which is too small for their initial design, then the tricks they use to "squeeze" the program into the core they have leads to great problems with errors and debugging. (See Fig. 1-3.) The influence of program run time on reliability has received little attention. It seems only obvious that if there is a very tight maximum on program run time (sometimes correctly and sometimes erroneously specified) that additional program tricks will be required which again contribute to the error and debugging problems. The implication is that not only will the debugging costs increase but that more sophisticated debug resistant errors may remain in the released version of the software to plague one during operation. Many people have observed that the effect of increased load on a system is to cause more frequent software errors. Most people using time-sharing systems feel that the mean-time between system crashes is strongly dependent on the number of users. (An interesting and definitive description of the software development process appears in Ref. 39.)

## 2.0 DEFINITION OF A SOFTWARE BUG

### 2.1 Problems in Definition

Definitions are always very difficult in the field of reliability, since we are trying to model a diffuse and complex physical situation by a mathematical model. The concepts of how to count multiple, repeated, and transient errors in hardware is most difficult. The author knows of cases where in a military contract, the contractor had to perform a reliability test of  $h$  hours on the equipment in question. If  $x$  or fewer failures occurred during the duration of the test, then the equipment passed. A board was created composed of government and contractor engineers. This failure board considered the test results and voted on each occurrence of abnormal behavior noted in the log book kept during the test to determine whether the occurrence should count as a failure. The most difficult case to decide on was where a certain logic card type failed  $m$  times. If this could be shown to be a simple repetition of the same transient failure, it would only count once, whereas if it was  $m$  separate failures, it would count  $m$  times. Often voting on the board was along strict "party lines," but often there were honest differences of engineering opinion.

If the definition of failure is difficult in the case of hardware where we have more experience and theoretical guidelines, it is even a more exacting task in the case of less well understood software. The author hopes the definitions proposed in the remainder of this section are a step toward a set of working definitions. At least they raise the salient issues in the reader's mind if he wishes to formulate his own definitions.

## 2.2 Definition of a Bug

The following definition of a software bug is proposed:

"One or more software bugs exist in a system if a software change is required to correct a single major error or minor error so as to meet specified or implied system performance requirements."

The following definitions are imbedded within the above definition:

**Change** - Any alteration (addition, deletion, correction) of the program code whether it be a single character or thousands of lines of code. Changes made to improve documentation or satisfy new specifications are important to record and study but are not counted as bugs.

**Major error** - A catastrophic event which interrupts or could interrupt most or all major system functions, e.g. as an infinite loop, system crash, a major memory overflow or data base corruption, etc.

**Minor error** - A marginal event which allows or could allow some portions of the system to operate properly while interrupting others, e.g. some missing output, some wrong output, an inaccurate computation, a recoverable transient error, etc.

**Specified performance requirements** - A written requirement, figure of merit, or parameter which qualitatively or quantitatively defines system performance.

**Implied system performance** - An unwritten requirement which is understood by the majority of the project team to be essentially equivalent to a written requirement.

Inherent in the above definitions and discussion is the assumption that errors can be and are detected and recorded. The detection of errors can be effected by monitoring the system (or simulated system) performance or by reading the code. Furthermore, it is assumed that each error is sufficiently well investigated so that it can be classified as hardware, software, operator, or unknown, and that the unknown category is small, say less than 20%.

## 2.3 Multiple Bugs and Changes

In this section we amplify on the previous definition in order to classify the cases of repetitious errors, multiple errors corrected by one change, and multiple changes to correct one error. We now introduce the concept of internal and external errors. An external error is a performance error of the system which is generally detected by executing the code. Theoretically, an external error could also be found by reading the code ("eyeballing"). An internal error is a coding error which is always found by reading the code either by man or machine.

The programmer could have initiated code reading due to one or more of the following factors:

- (1) an external error has been detected, and he is trying to find the corresponding internal error.
- (2) he was reading the code to verify a program change before submitting it to the computer.
- (3) he was performing a code reading test to detect errors.
- (4) a colleague told him of an actual or potential error.

We may think of internal errors as causes and external errors as effects. Thus, if a single internal error results in an associated single external error, we call it a single bug. If an internal error results in a minor or no detectable external error, then no bug exists. If an external error exists, and we are sure it is a software problem, then a bug exists regardless of whether or not we can find the corresponding internal error.

One important class of external errors for which no internal error can be found are transient errors. These exist for too short a time for isolation of the cause. A transient error which is found is no longer transient. If before the internal error is found, the transient occurs  $m$  times, it is still only counted as a single error if the symptoms are the same.

Theoretically, many internal errors can combine to cooperatively cause one external error. We expect that this is an event with a low probability of occurrence. Since only a single external error exists, this would be classified as one bug.

A more common multiple error case would be where one internal error causes  $m$  external errors. Initially, if the  $m$  external errors are known, but no corresponding internal error has been found, then we classify this result as  $m$  bugs. At a later time, several days or weeks hence, when the common internal bug is found, we decide to reclassify this result as a single bug. Thus, if we are recording the cumulative number of bugs frequently, say each day, then the above event would first count as an increase of  $m$  bugs, and later when the single common internal error was found, there would be a decrease of  $m-1$  bugs. Of course, if the data were taken less frequently, the diagnosis of the internal error and the external error would fall within the same time interval and only the net result, i.e. one bug, would be recorded. In any event we could probably treat the number of bugs as defined herein as an upper bound if multiple external errors occur frequently.

## 2.4 Old or New Bugs

It is useful in trying to model the dynamics of the debugging process to know whether a bug is old or new. To be more precise we might use the terminology previously corrected bugs and generated bugs.

A previously corrected bug is one which reoccurs in substantially the same form after the programmer terminated his work on a code change believing that the error was corrected. A conclusive decision that a bug was a previously corrected one can only be made based on an internal error diagnosis.

A generated error is one which did not exist until it was created as a by-product of a code change made to correct some bug. A generated error is usually best diagnosed by finding an internal error. However, it is sometimes possible to base such a classification on an external error, i. e. if a newly created variable appears in the wrong output form. Of course, all the above definitions rely on subjective judgment of the programmers. However, it is hoped that qualified personnel could use these definitions to obtain repeatable results.

## 3.0 DEFINITION OF SOFTWARE RELIABILITY

### 3.1 Factors in the Definitions

In the early days of hardware reliability there was much major soul searching and thrashing about until a widely accepted definition of reliability was formulated. In the preceding section some working definitions of software bugs were developed. In this section we will attempt to develop a working definition of software reliability.

Our experience in the hardware area has taught us that reliability must be defined as the probability that some event occurs over a period of time (operating time). In our case the event is success of the software, i. e. error free software operation. We are now helped by the definitions of the previous section in defining the event error free software operation. Error free software operation over the operating time interval 0 to  $t$  means that no software bug (external software error) occurred over that interval. Clearly, if we are to define an external error, we must define what constitutes successful performance.

Another factor which must be specified is the hardware environment. This suggests certain obvious factors as well as more subtle ones. Obviously, a FORTRAN program written for use on an IBM computer will probably need modification before it can be used on a UNIVAC computer. We find more subtle differences when a computer is being specially designed for a project and the software and hardware are to be mated on the laboratory prototype. Often there are last minute changes which make for significant differences between the prototype and the field installation. Also, as an economy measure, the prototype may be a smaller configuration than the actual field installation. We are now faced with the situation where the final tests on the hardware complex will be carried out on a computer which differs from the operational model.

Finally, we must be concerned about the intended task the system must perform. Although there are always extensive efforts to write comprehensive specifications, problems occur. For example, returning to the previously cited APOLLO program, it was decided that the astronauts would never enter a call for the wrong program during the orbit phase of flight so no error checking for program calls was incorporated in the software. Subsequently, an astronaut while in orbit called for the ground initialization and alignment program. Would you call this a software error?

In many cases a system is originally sized for a particular data input rate. As the system begins to function successfully in the field after elimination of the initial bugs, there is a trend to employ it more widely. The error rate of the system increases as the data input rate increases. A classic example of this effect is the initial deployment of the SABER Airlines Reservation System. Every time a new group of terminals was added to the system from a new city or group of cities a new crop of bugs was encountered. The system was allowed to grow, in stages, at a controlled rate by establishing an upper failure rate limit and a lower failure rate limit. When debugging removed enough errors so that the system reached the lower failure rate limits, new terminals were added until the upper limit was reached. As soon as the debugging team reduced the failure rate to the lower limit, the cycle was repeated.

### 3.2 Definition

A definition for software reliability is given below in keeping with the factors discussed above. This definition is a slight modification of one given by Hesse<sup>8</sup>:

"Software reliability is defined as the probability that a given software program operates for some time period, without an external software error, on the machine for which it was designed given that it is used within design limits."

Once we have related reliability to a probability, as in the above definition, the mathematical basis of the measure is well founded. Of course, the problems in interpreting terms such as external error and design limits still exists.

### 3.3 Reliability Theory

The following brief development of the reliability function, hazard function, and mean time to failure is included for those unfamiliar with reliability theory<sup>9</sup>. We begin with the standard probability functions



$$R(t) = P(\underline{t} > t) \quad (3-1)$$

$$F(t) = 1 - R(t) \quad (3-2)$$

$$f(t) = \frac{dF(t)}{dt} = - \frac{dR(t)}{dt} \quad (3-3)$$

where

$\underline{t}$  = the random variable time to failure.

$t$  = a particular value of the random variable.

$R(t)$  = the reliability function, which yields the probability of no failure in the interval 0 to  $t$ .

$F(t)$  = the cumulative distribution function, which yields the probability of failure in the interval 0 to  $t$ .

$P(\underline{t} > t)$  = the probability that the time to failure lies outside the interval 0 to  $t$ .

Workers in the field of reliability have found it convenient to define a different conditional probability function called the failure rate or hazard function,  $z(t)$ . One may define  $z(t)$  in a manner analogous to the definition of  $f(t)$  in terms of the probability that a failure occurs in the interval  $t$  to  $t + \Delta t$

$$\left\{ \begin{array}{l} \text{Probability of a failure} \\ \text{in the interval } t \text{ to } t+\Delta t. \end{array} \right\} = P(t < \underline{t} < t+\Delta t) = f(t)\Delta t \quad (3-4)$$

$$\left\{ \begin{array}{l} \text{Probability of a failure} \\ \text{in the interval } t \text{ to } t+\Delta t \\ \text{given the fact that failure} \\ \text{did not occur prior to } t. \end{array} \right\} = P(t < \underline{t} < t+\Delta t | \underline{t} > t) = z(t)\Delta t \quad (3-5)$$

From these definitions it can be shown that

$$z(t) = \frac{f(t)}{R(t)} = - \frac{1}{R(t)} \frac{dR(t)}{dt} \quad (3-6)$$

Solving this differential equation for  $R(t)$  subject to the initial condition that the item is initially good, i. e.,  $R(t=0) = 1$  yields

$$R(t) = e^{-\int_0^t z(x) dx} \quad (3-7)$$

Another measure which is often used is the mean time to system failure, MTTF. This is simply given as the first moment of the random variable  $\underline{t}$ .

$$\text{MTTF} = \int_0^{\infty} t f(t) dt \quad (3-8)$$

It can be shown that Eq. 3-8 can be reduced to the simpler computational form

$$\text{MTTF} = \int_0^{\infty} R(t) dt \quad (3-9)$$

For the simple case where  $z(t)$  is a constant Eqs. (3-7) and (3-8) yield

$$z(t) = \lambda \quad (3-10)$$

$$R(t) = e^{-\lambda t} \quad (3-11)$$

$$\text{MTTF} = 1/\lambda \quad (3-12)$$

#### 4.0 ERROR DATA AND MODELS

##### 4.1 Introduction

When one attempts to apply probability and statistics to an engineering problem, two approaches immediately suggest themselves. The pure statistical approach is to define the variables and performance measure(s) and construct and carry out an experiment. The experimental results are statistically analyzed to determine quantitatively what relationships (if any) exist among the performance measure(s) and the variables. The other approach is to formulate a probabilistic hypothesis about how the variables interact and write a corresponding equation relating the performance measure(s) to the variables. Based on these models, experiments are planned to verify the hypotheses and to determine the constants in the equations. The reader may wish to view these approaches as analogous to the distinct approaches of a theoretical and an experimental physicist. In either case it is appropriate to begin by examining the data available in the literature at the outset.



The first relevant question is what data should we examine. If we ask experienced software managers what quantitative measures they use to gauge the progress of a software program, they answer, none or refer to graphs of the cumulative number of errors removed from the software. Those who believe this is a significant measure look for the slope of the curve to approach zero before deciding the software is sufficiently debugged for release. This section of the paper discusses some of the sparse experimental data available in the open literature on the number of errors removed from a computer program. The next section builds upon this data and certain hypotheses to evolve a probabilistic error model.

Also of interest are the error types and frequencies of occurrence. Unfortunately, only a small amount of data has been accumulated in this area, and until the gross models (such as those discussed above which treat all errors alike) have been verified, such refinement in a model is perhaps unjustified.

#### 4.2 Post Release Data

As is the case with all studies, data is difficult and generally costly to obtain. Good and complete records are not kept in most situations. Record keeping is generally better in the case of large military and space programs and after a release of a large commercial operating system. Although a fairly large amount of such data exists, military secrecy and industrial proprietary policies inhibit its publication in many cases. Some of this data which has been published appears in Refs. 8 and 10.

Assume that a typical operating system for a large computer is undergoing continual development and that new features and capabilities are being added. The manufacturer's development group deals with a continually changing product, but external versions (generally called releases) are only made available periodically, say every 6 months. Although the manufacturer tries to thoroughly test each release, the exercising of the program by a fair proportion of the large and diverse user community is more comprehensive than any test he can devise. Consequently, soon after release of a new version, the number of errors found per month (error rate) rises rapidly to a peak. As these are diagnosed and corrected, the number of residual errors decreases and the error rate begins to decrease. When a new release is distributed, this behavior is repeated. Such typical behavior is sketched in Fig. 4-1. Note that the vertical axis is normalized by dividing by the total number of machine language instructions. This should allow us to compare both large and small programs to see if there is a behavior pattern independent of size. Detailed data on the normalized number of errors since release for three different supervisory systems (operating systems) is given in Fig. 4-2. Note that the horizontal axis units are months of debugging  $\tau$ . In this case  $\tau$  is identical with operating time,  $t$ ; however, this is not always the case.

Note that the shapes depicted in Fig. 4-2a, b, c vary. If we assume that the number of remaining errors decreases monotonically and that the error discovery rate is proportional to number of remaining errors, exponential decay is obtained. This explains in a gross way the "tail" of the curves. The initial behavior may be due to the fact that initially only a few installations are using the new release, and it is not until a few months later that a sizeable proportion of users have instituted this software. Thus, it might be more appropriate to let  $\tau$  represent a more general resource variable such as user-months, or to serve as a more realistic horizontal scale parameter. (See Ref. 10 for a more detailed discussion of these curve shapes.)

In Fig. 4-3 the error rate curves for four applications programs are presented. In this case the origin  $\tau = 0$  represents the start of program integration where all the individual modules of code are put together to form a system. As is well known, at this point incompatibilities between the modules crop up and a new set of interface errors must be debugged. We may employ the same argument used previously to describe the tails of the curves in Fig. 4-3. Also, if we think of  $\tau$  as a general resource variable which is a function of man-hours of debugging and computer test hours, this may explain the initial behavior. Again, no data is available to test this hypothesis.

#### 4.3 Error Model

Referring to the data discussed in the previous section we see that although the curve shapes differ, the vertical and horizontal scales are similar. Based on this result we can proceed to formulate a general error model using the number of machine language instructions as a normalizing factor.\*

Basically, the error model used in this paper assumes that the total number of errors in the program is fixed and that if we record the cumulative number of errors corrected during debugging, then the difference represents the remaining errors. The following section on reliability models will relate the probability of encountering a software bug to the number of residual bugs.

The normalized error rate is defined as\*

$$\rho(\tau) = \frac{\text{errors}}{\text{total number of instructions}} / \text{month of debugging time.} \quad (4-1)$$

Thus, Figs. 4-1, 4-2, and 4-3 are plots of  $\rho(\tau)$  vs.  $\tau$ .

Since we are interested in the total number of errors removed, we will define a cumulative error curve,  $e(\tau)$ , which is the area under the  $\rho(\tau)$  curve:

$$e(\tau) = \int_0^{\tau} \rho(x) dx = \frac{\text{cumulative errors}}{\text{total number of instructions}} \quad (4-2)$$

\*Recent work (see Ref. 40, 41) suggests that instead of the total number of instructions, a better measure of program length is the total number of operators and operands in the program.

and  $\rho(\tau)$  is of course the slope of the  $\epsilon(\tau)$  curve:

$$\rho(\tau) = d\epsilon(\tau)/d\tau \quad (4-3)$$

A curve of the cumulative error data for the supervisory system A of Fig. 4-2 is shown in Fig. 4-4. If similar curves for  $\epsilon(\tau)$  were drawn for the other examples of Figs. 4-2 and 4-3, all would start at zero, increase slowly, then move rapidly, and finally, more slowly approaching a slowly increasing or zero rate. Because of this similarity in behavior, a cumulative curve such as  $\epsilon(\tau)$  is not too useful to depict differences in behaviors; thus, the derivative curve,  $\rho(\tau)$ , is more useful for this purpose. Both curves are needed for a detailed study.

If we assume that the total number of errors in the program  $E_T$  is constant and that the program contains  $I_T$  total instructions and that no new errors are added during debugging,\* then the asymptote which the  $\epsilon(\tau)$  curves approach is  $E_T/I_T$ . If we assume that all detected errors are corrected errors, then by inspection of Fig. 4-4, we can write an expression for the number of residual errors:

$$\epsilon_r(\tau) = (E_T/I_T) - \epsilon_c(\tau) \quad (4-4)$$

We assume that in any sizeable program it is impossible to remove all errors, so

$$\epsilon_c(\tau) < E_T/I_T \quad (4-5)$$

$$\epsilon_r(\tau) > 0 \quad (4-6)$$

Also since we assume that most programs eventually reach a reasonable debugged state, we may assume that for large  $\tau$ ,  $\epsilon_r(\tau) I_T/E_T$  is small.

In order to test the hypothesis that the normalized behaviors of  $\epsilon_c(\tau)$  and  $\rho(\tau)$  hold for a wide variety of program sizes we make the following comparisons with the data in Figs. 4-2 and 4-3: (1) In order to test the hypothesis that the normalized number of errors  $E_T/I_T$  is somewhat constant for a variety of programs, we compute the ratio and compare the results. (2) An allied hypothesis is that debugging proceeds at a roughly similar average rate  $\rho_0$  over an entire project. The results are given in Table 4-1. The value of  $E_T/I_T$  varies about the average by +48% and -31% and that of  $\rho_0$  varies about the average by +75% and -31%. Note that all these programs are about 1/4 million machine language statements in size. The data is often "dirty" since in some projects only program corrections are counted; whereas, in others specification and improvement changes are lumped in with actual error changes.

Furthermore, the applications programs presented debugging information during the program integration phase of software development; whereas, the supervisory programs reported errors after release. It is not unreasonable that the errors found during system integration and after release of a large software package are roughly commensurable. Based on three software programs, it has been shown that the ratio of changes after release to changes during integration and test was about 0.8.<sup>30</sup> If we compare the average value of  $E_T/I_T$  for the supervisory programs to that for the application programs in Table 4-1, the ratio is about 0.7. Based on the above factors the data appears to verify the hypothesis that  $E_T/I_T$  and  $\rho_0$  are approximately constant for similar size programs. We now present similar data for small programs<sup>31</sup> in Table 4-2. In this case both the values of  $E_T/I_T$  and  $\rho_0$  vary about the average by +79% and -36%.

The data in Table 4-2 includes data taken during module test as well as during integration testing and as might be expected (because of the two phases being lumped), the average value of  $E_T/I_T$  for the small programs data is 2.15 times larger than the large program data whereas the value of  $\rho_0$  is 1.53 times larger. Drawing these various facts together allows us to state that within a factor of perhaps 2, the values of  $E_T/I_T$  and  $\rho_0$  seem to be constant for a wide variety of programs. Furthermore, within a similar factor, the number of bugs per machine instruction found during module test, integration testing, and after release are roughly the same.

One further comment is in order before we leave the subject of error models. Some experienced programmers have challenged the assumption that no new errors are generated during debugging. In Fig. 4-5 three dynamic debugging behaviors are illustrated. In Fig. 4-5a no new errors are added, and the situation depicted is just the one which we have been discussing. In Fig. 4-5b errors are added; however, the removal rate exceeds the generation rate and equilibrium is obtained. If the number of errors added is small percentage wise, even cases (a) and (b) are approximately numerically equivalent. Fig. 4-5(c) depicts a case where the error generation rate exceeds the error removal rate and the process diverges. A newly devised model, formulated by this author and his co-workers describes error generation in cases (b) and (c), and is discussed and developed in Reference 34.

## 5.0 RELIABILITY MODELS

### 5.1 Introduction

In order to formulate a reliability model, one can take a microscopic or a macroscopic approach. In the microscopic approach we would try and identify individual bugs (either deterministically or probabilistically), the type of bug, the path in the program, and how frequently the path is traversed. Initial attempts along these lines have convinced this author that such an approach, while necessary in the long run involves a more detailed knowledge of program structure and bug types than is now available. The macroscopic approach where all bugs are lumped and treated equally will be employed here. The validity of the result depends on considerable 'averaging' occurring in a large program.

\* This model has recently been extended to include error generation terms. (See Ref. 34.)

## 5.2 Basic Assumptions

We assume that operational software errors occur due to the occasional traversing of a portion of the program in which a hidden software bug is lurking. We begin by writing an expression for the probability that a bug is encountered in the time interval  $\Delta t$  after  $t$  successful hours of operation. This must be proportional to the probability that any randomly chosen instruction contains a bug, i.e., the fractional number of remaining bugs  $e(\tau)$ .

From a study of basic probability and reliability theory,<sup>9</sup> we learn that the probability of failure in time interval  $t$  to  $t + \Delta t$  given that no failures have occurred up till time  $t$  is proportional to the failure rate (hazard function  $z(t)$ ).

$$P(t < t_f \leq t + \Delta t | t_f > t) = z(t)\Delta t = K e_r(\tau)\Delta t \quad (5-1)$$

where  $t_f$  = operating time to failure, (occurrence of a software error)

$P(t < t_f \leq t + \Delta t | t_f > t) \equiv$  probability of failure in interval  $\Delta t$ , given no previous failure.

$K$  = an arbitrary constant\*

Note that in Eq. 5-1 two time variables appear: first there is  $t$  the operating time in hours of the system and second there is  $\tau$  the debugging time in months (or more generally, the debugging resource variable). Once the assumptions in Eq. 5-1 have been made, the reliability and mean time to failure functions follow directly.

## 5.3 Reliability Model

By combining Eqs. 5-1 and 3-7 and assuming that  $K$  and  $e_r(\tau)$  are independent of operating time  $t$ , we obtain for the reliability function

$$R(t) = e^{-[K e_r(\tau)]t} = e^{-\gamma t} \quad (5-2)$$

Basically the above equation states that the probability of successful operation without software bugs is an exponential function of operating time. When the system is first turned on,  $t = 0$  and  $R(0) = 1$ . As operating time increases the reliability monotonically decreases as shown in Fig. 5-1. We depict the reliability function for three values of debugging time,  $\tau_0 < \tau_1 < \tau_2$ . From this curve we may make various predictions about the system reliability. For example, looking along the vertical line  $t = 1/\gamma$  we may state:

1. If we spend  $\tau_0$  hours of debugging, then  $R(1/\gamma) = 0.35$
2. If we spend  $\tau_1$  hours of debugging, then  $R(1/\gamma) = 0.50$
3. If we spend  $\tau_2$  hours of debugging, then  $R(1/\gamma) = 0.75$

## 5.4 MTTF Model

A simpler way to summarize the results of the reliability model is to compute the mean time to (software) failure, MTTF by substituting Eq. 5-2 into Eq. 3-9.

$$\text{MTTF} = \frac{1}{K e_r(\tau)} \quad (5-3)$$

If we let  $\rho(\tau)$  be modeled by a constant rate of error correction  $\rho_o$  (see Ref. 10 for other models), then solution of Eqs. 5-3 and 4-4 yields

$$\text{MTTF} = \frac{1}{K \left[ \frac{E_T}{I_T} - \rho_o \tau \right]} = \frac{1}{\beta (1 - \alpha \tau)} \quad (5-4)$$

where  $\beta = \frac{E_T}{I_T} K$  and  $\alpha = \frac{\rho_o I_T}{E_T}$

In Fig. 5-2,  $\beta \times \text{MTTF}$  is plotted vs.  $\alpha \tau$ . We see that the most improvement in MTTF occurs during the last 1/4 of the debugging.

## 5.5 Other Models

Other similar models have been proposed in the literature. Jelinski and Moranda<sup>13</sup> propose a hazard function of the form

\* In earlier work (see Ref. 10 or 12) an attempt was made to achieve a more micromodel by splitting  $K$  into two factors,  $K'$  an arbitrary constant, and  $r_p$  the instruction processing rate. This elaboration is not included here since, to date, no data has been obtained to define or calculate  $r_p$ .



$$z(\tau) = \phi [N - (i-1)] \quad (5-5)$$

where:  $\phi$  = Constant of proportionality.

$N$  = Total number of errors present.

$i$  = Number of errors found by debugging time  $\tau_1$ .

Comparison of Eq. 5-5 with Eqs. 5-1 and 4-4 shows them to be identical if

$$E_T = N \quad (5-6)$$

$$\frac{K}{I_T} = \phi \quad (5-7)$$

$$\epsilon_c(\tau) = \frac{i-1}{I_T} \quad (5-8)$$

Equations 5-6 and 5-7 are merely notational differences and Eq. 5-8 is nearly the same (i.e., would be identical if  $\epsilon_c(\tau) = i/I_T$ ).

In another paper Shick and Wolverson<sup>14</sup> modify Jelinski and Moranda's model and assume that the failure rate is proportional to the number of remaining errors and increases with operating time  $t$

$$z(t) = \phi [N - (i-1)]t \quad (5-9)$$

One rationale for postulating an increase in  $z(t)$  would be if operation were viewed as a succession of different trials which gradually closes in on the remaining errors (sampling without replacement). However, one could argue to the contrary that  $z(t)$  should decrease with  $t$ , since the latter errors are the subtle ones which take a long while to encounter in operation. The author believes that in most cases of large, intricate, well tested, real-time systems the hazard will remain constant once the initial field debugging of a new release is finished. The small number of subsequent patches generated should not be significant. Failure should be caused by rare combinations of input data and path traversals, with the time between failures governed by an exponential distribution, yielding a constant hazard. Experimental data is necessary to choose among these hypotheses.

Other related reliability and error models are discussed in Refs. 15, 16 and 17. The author has also devised a micro reliability model where the failure rate is related to the number of paths in the program, the path traversal rate, the path run time, and the probability of error along the path. (See Ref. 42) We now turn in the next two sections to a discussion of how we can experimentally measure reliability and use these measurements in conjunction with the models of this section to determine the unknown model parameters  $K$  and  $E_T$ .

## 5.6 Experimental Reliability Data

If we had just deployed a large hardware-software system for field use, we could monitor its reliability by carefully recording the operating time and documenting each failure in detail. Thus, we could obtain the times between failure. Investigation of each failure should allow one to classify all failures as hardware, software, operator, or unknown. If we segregate the software times between failure and plot their average week by week, we will have a quantitative measure of operational software reliability. We would expect the operational MTTF to increase for the first month (year, in some cases) or so as software bugs detected in service are removed, then gradually to level off to a relatively constant value. This is, of course, an after-the-fact evaluation of the software design and does not allow one to measure progress and/or need for improvement of the software design while it is under development.

The earliest stage at which an entire system can be functionally tested is during system integration using the system exerciser (functional test) program. If this test is performed at the beginning of system integration, the result will be a succession of very short runs and immediate crashes. Most software test personnel would instinctively comment that this is as expected since the system is still in "poor shape" and such a test should be delayed until the end when the system is in "good shape". A bit of reflection leads one to the conclusion that it is just this frequent crashing which leads to a quantitative assessment of the poor initial reliability.

We now focus on the test data and how it should be analyzed. The necessary information which must be recorded for each run of the system test program is how long the test ran, whether an error occurred, and if the error is a software error. Sufficient dumps and other documentation must be recorded for subsequent analysis in order to segregate errors into hardware, software, operator, etc., errors. Each of the  $r$  successful runs represent  $T_1, T_2, \dots, T_r$  hours of success. If there are  $n$  total runs, then each  $(n-r)$  unsuccessful run represents  $t_1, t_2, \dots, t_{n-r}$  successful run hours before failure. The total number of successful run hours  $H$  is given by

$$H = \sum_{i=1}^r T_i + \sum_{i=1}^{n-r} t_i \quad (5-10)$$

Assuming that the failure rate is constant, we denote it by  $\lambda$  and compute it as the number of failures per hour



$$\text{Failure Rate} = \lambda = \frac{n-r}{H} \quad (5-11)$$

The MTTF for a constant failure rate is the reciprocal (see Eq. 3-12) of the failure rate

$$\text{MTTF} = \frac{1}{\lambda} = \frac{H}{(n-r)} \quad (5-12)$$

Now Eqs. 5-11 and 5-12 represent the total system failure rate and MTTF. Since we are mainly interested in software failures, we assume that the outputs as well as dumps are carefully investigated for the  $(n-r)=x$  failures. Based on the above analysis, the failures are divided into  $x_h$  hardware failures,  $x_s$  software failures,  $x_o$  operator failures, and  $x_u$  unknown failures. Hopefully, the unknown ratio  $x_u/x$  will be 25% or smaller so that most of the data is classifiable.

Then the software failure rate and MTTF are defined by

$$\lambda_s = \frac{x_s}{H} \quad (5-13)$$

$$\text{MTTF}_s = \frac{H}{x_s} \quad (5-14)$$

Thus, based on the results on this occasional test we can plot  $\lambda_s$  and  $\text{MTTF}_s$  vs.  $\tau$  the debugging time. Such charts should allow a quantitative measure of the progress in improving software quality. Thus, after  $\tau_a$  hours of debugging we would have a measure of MTTF and  $R(t)$  and by extrapolation of the curves we could predict MTTF and  $R(t)$  after  $\tau_b > \tau_a$  months of debugging. Unless we knew the functional form of the variations in  $R(t)$  and MTTF with  $\tau$  and could determine an appropriate extrapolation scheme, accurate predictions would be limited to small excursions into the future.

### 5.7 Estimation of Model Constants

Rather than use the raw experimental data and extrapolation for prediction, we can assume an underlying model for  $\lambda_s$  and  $\text{MTTF}_s$  and use the data to estimate the model parameters. If the hypothesized model is correct, then predictions using this technique should be superior to the extrapolation technique of the previous section.

For the software reliability model defined in Sections 5-2 and 5-3

$$R(t, \tau) = \exp \left[ -K \left( \frac{E_T}{I_T} - \epsilon_c(\tau) \right) t \right] \quad (5-15)$$

$$\text{MTTF}(\tau) = \frac{1}{K \left( \frac{E_T}{I_T} - \epsilon_c(\tau) \right)} \quad (5-16)$$

Note that if we assume a known program size and careful collection of error data, then  $I_T$  and  $\epsilon_c(\tau)$  are known values and only the constants  $K$  and  $E_T$  remain to be determined. These two unknowns  $K$  and  $E_T$  can be evaluated by running a functional test after two different debugging times,  $\tau_1 < \tau_2$  chosen so that  $\epsilon_c(\tau_1) < \epsilon_c(\tau_2)$ . We then equate Eqs. 5-14 and 5-16 at times  $\tau_1$  and  $\tau_2$

$$\frac{H_1}{x_{s1}} = \frac{1}{K \left[ \frac{E_T}{I_T} - \epsilon_c(\tau_1) \right]} \quad (5-17)$$

$$\frac{H_2}{x_{s2}} = \frac{1}{K \left[ \frac{E_T}{I_T} - \epsilon_c(\tau_2) \right]} \quad (5-18)$$

Taking the ratio of Eq. 5-17 to 5-18 and using Eq. 5-14 yields

$$\hat{E}_T = \frac{I_T \left[ \left( \lambda_{s2} / \lambda_{s1} \right) \epsilon_c(\tau_1) - \epsilon_c(\tau_2) \right]}{\left( \lambda_{s2} / \lambda_{s1} \right) - 1} \quad (5-19)$$

Once  $\hat{E}_T$  has been computed from Eq. 5-19, we obtain  $\hat{K}$  by substituting Eq. 5-19 into 5-17 which yields

$$\hat{K} = \lambda_{s_1} / [(\hat{E}_T / I_T) - \epsilon_c(\tau_1)] \quad (5-20)$$

The "hats" above  $E_T$  and  $K$  in Eqs. 5-19 and 5-20 denote estimates of the parameter. Note that if there was no debugging between  $\tau_1$  and  $\tau_2$  so that  $\epsilon_c(\tau_1) = \epsilon_c(\tau_2)$ , the numerator of Eq. 5-19 becomes zero, i. e., Eqs. 5-17 and 5-18 are no longer independent and the estimate fails.

Further discussions of this parameter estimation technique, the more powerful maximum likelihood estimation technique, and accuracy questions appear in Ref. 18.

## 6.0 ALLIED AREAS

This paper has concentrated on one aspect of software reliability, its measurement theoretically and experimentally. There are many other allied areas relating to this subject: system recovery techniques<sup>21</sup>, program design for low error content<sup>22,23</sup>, the production of standard computational programs which are very reliable<sup>29</sup>, structural complexity,<sup>44,45</sup> etc.

An historical account of the SAGE air defense system, one of the first real time hardware-software systems is worth reading<sup>24</sup>. Other material of interest appears in the Record of the 1973 IEEE Symposium on Computer Software Reliability<sup>15</sup>, the Proceedings of the Brown Symposium,<sup>10</sup> the NATO Conference on Software Engineering<sup>28</sup>, the book edited by Rustin<sup>25</sup>, the Proceedings of recent conferences on Software Reliability (Ref. 33, 35, 36, 46, 47, 48) and a newly published journal<sup>32</sup>. More material is appearing each month in the computer research journals on this dynamic new field.

## REFERENCES

1. Harold Sackman, "Computers, System Science, and Evolving Society," John Wiley and Sons, New York, N. Y., 1969.
2. Terry M. Walker and William W. Cotterman, "An Introduction to Computer Science and Algorithmic Processes."
3. United States Air Force Report, "Information Processing/Data Automation Implications of Air Force Command and Control Requirements in the 1980's," (CCIP-85), Vol. 1, SAMSO/XRS-71-1, Apr. 1972.
4. Arthur C. Clarke, "2001 A Space Odyssey," based on the MGM film screenplay of the same name. Signet Books, New York, N. Y., 1968.
5. M. Hamilton, MIT Charles Stark Draper Labs., Cambridge, Mass., Private Communication.
6. Barry W. Boehm, "Software and Its Impact: A Quantitative Assessment," Datamation Magazine, May 1973.
7. E. Yourdon, "Reliability Measurements for Third Generation Computer Systems," Proceedings of the 1972 Annual Reliability Symposium, IEEE, New York, N. Y.
8. J. Hesse, A. Kientz, J. Dickson and M. Shooman, "Quantitative Analysis of Software Reliability," 1972 Annual Reliability Symposium Proceedings, IEEE, New York, N. Y., January 1972.
9. M. Shooman, "Probabilistic Reliability: An Engineering Approach," McGraw-Hill Book Co., New York, N. Y., 1968.
10. M. Shooman, "Probabilistic Models for Software Reliability Prediction," Conference on Statistical Methods for the Evaluation of Computer System Performance, Brown Univ., Nov. 1972. Published in "Probabilistic Models for Software," Freiburger Editor, Academic Press, New York, N. Y., 1972, pp. 485-502.
11. Fumio Akiyama, "An Example of Software System Debugging," IFIP Congress 71, Ljubljana, Yugoslavia, Aug. 1971.
12. M. Shooman, "Probabilistic Models for Software Reliability Prediction," 1972 International Symposium on Fault-Tolerant Computing, Newton, Mass., June 21, 1972, IEEE Computer Society.
13. J. Jelinski and P. B. Moranda, "Software Reliability Research," same source as Reference 10.
14. G. J. Schick and R. W. Wolverton, "Assessment of Software Reliability" 11th Annual Meeting, German Operations Research Society, Hamburg, Germany, September 1972.
15. E. Girard and J-C. Rault, "A Programming Technique for Software Reliability," Record of the 1973 IEEE Symposium on Computer Software Reliability, New York, N. Y., May 1973, p. 44.
16. B. Littlewood and J. L. Verall, "A Bayesian Reliability Growth Model for Computer Software," Same source as Ref. 15, p. 70.
17. J. Jelinski and P. B. Moranda, "Applications of a Probability-Based Model to a Code Reading Experiment," Same source as Ref. 15, p. 78.

18. M. Shooman, "Operational Testing and Software Reliability Estimation During Program Developments," Same source as Ref. 15, p. 51.
19. Bell System Technical Journal, Dec. 1970. Articles on testing of TSPS No. 1 and ESS No. 1 ADF.
20. Donald E. Knuth, "An Empirical Study of FORTRAN Programs," Computer Science Department Report No. CS-186, Stanford University, 1970.
21. E. Yourdon, "Design of On-Line Computer Systems," Prentice-Hall, Inc., Englewood Cliffs, N. J., 1972, p. 515.
22. G. Weinberg, "The Psychology of Computer Programming," Van Nostrand Reinhold Co., New York, N. Y., 1971.
23. O. Kahl, E. Dijkstra, C. Hoare, "Structured Programming, Academic Press, London, 1972.
24. H. Sackman, "Computers, System Science, and Evolving Society," John Wiley and Sons, New York, N. Y., 1967.
25. R. Rustin, "Debugging Techniques in Large Systems," Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.
26. W. Freiburger, "Statistical Computer Performance Evaluation," Academic Press, New York, NY, 1972.
27. H. Sackman, "Man-Computer Problem Solving: Experimental Evaluation of Time-Sharing and Batch Processing," Auerbach Publishers, Philadelphia, Pa., 1970.
28. NATO Conferences: (1) "Software Engineering Techniques," (Eds. Burton, J. N. and B. Randell), 1969; (2) "Software Engineering," (Eds. Naur, P. and B. Randell) Garmisch, Germany, Oct. 1968.
29. W. Cowell, "Proceedings of the Software Certification Workshop," Aug. 1972, Argonne National Laboratory, Argonne, Ill.
30. M. Hyman, IBM Federal Systems Division, Morris Plains, N. J., Private Communication.
31. M. Shooman, "Software Reliability: Analysis and Prediction," published in "Generic Techniques in Systems Reliability Assessment", Ed. by E. J. Henley and J. W. Lynn, Noordhoff Intl. Pub., Reading, Mass., 1976.
32. IEEE Trans. on Software Engineering, IEEE Computer Society, New York City, (First published in 1975.) Paper by John Musa, "A Theory of Software Reliability and Its Applications," Vol. SE-1, No. 3, Sept. 1975.
33. "Proceedings of the Intl. Conference on Reliable Software," Los Angeles, April 21-23, 1975, IEEE Cat. No. 75 CH0940-7CSR.
34. M. L. Shooman and S. Natarajan, "Effect of Manpower Deployment and Error Generation on Software Reliability," Proceedings of the MRI Symposium on Computer Software Reliability, April 1976. Available from Polytechnic Press, Polytechnic Institute of New York.
35. "Proceedings of the Software Certification Workshop," Gramby, Colorado, sponsored by NSF. (Proceedings available from Dr. Wayne Cowell, Argonne Natl. Labs., 9700 S. Cass Ave., Argonne, Ill. 60439.
36. "Proceedings of a Symposium on the High Cost of Software," Monterey, Calif., Sept. 17-19, 1973, sponsored by AFOSR, ARO, ONR. (Proceedings available from Jack Goldberg, Stanford Research Institute, Menlo Park, Calif. 94025.)
37. S. J. Amster and M. L. Shooman, "Software Reliability: An Overview," Reliability and Fault Tree Analysis, p. 655, ed. R. Barlow et al., Society for Industrial and Applied Math., Philadelphia, Pa., 1975.
38. R. W. Wolverton, "The Cost of Developing Large Scale Software," International Convention, Mar. 20-23, 1972, IEEE, New York.
39. F. P. Brooks, "The Mythical Man-Month: Essays on Software Engineering," Addison-Wesley, Reading, Mass., 1975.
40. M. Halstead, "Software Physics: Basic Principles", IBM Research Report RJ1582, IBM Research, Yorktown Heights, N. Y., May 1975.
41. A. Laemmel and M. Shooman, "Statistical (Natural) Language Theory and Computer Program Complexity," Internal Report, Polytechnic Institute of New York, Winter 1976.
42. M. Shooman, "Structural Models for Software Reliability Prediction," p. 268, Proceedings 2nd Intl. Conf. on Software Engineering, Oct. 13-15, 1976, IEEE Computer Society, New York.
43. L. M. Putnam, "A Macro-Estimating Methodology for Software Development," p. 138, Digest of Papers, Thirteenth IEEE Computer Society Intl. Conference, Sept. 7-10 1976, IEEE, N. Y.
44. T. J. McCabe, "A Complexity Measure," Proceedings, 2nd Intl. Conference on Software Engineering, Vol. 2, Oct. 13-15, 1976, IEEE, New York, p. 69.

45. M. L. Shooman, "Software Engineering: Reliability, Design, Management," Notes for Course EE909, Polytechnic Institute of N. Y., 1976.
46. Proceedings of the 1st Intl. Conference on Software Engineering, IEEE Computer Society, Sept. 11-12, 1975, New York.
47. Proceedings, 2nd Intl. Conference on Software Engineering, Oct. 13-15, 1976, IEEE Computer Society, New York.
48. M. L. Shooman, "Models, Data, and Analysis - A Summary of the MRI Symposium," Proceedings, 2nd Intl. Conference on Software Engineering, Oct. 13-15, 1976, IEEE Computer Society, New York.

TABLE 1-1

Number of Computer Installations in the United States  
(actual and estimated\*)

Year	Total Installations	
	Source A	Source B
1945	1	-
1950	20	10-15
1955	1,000	1,000
1960	6,000	6,000
1965	30,000	30,000
1970	60,000*	50,000*
1975	85,000*	80,000*

Source A. Ref. 1, p. 29

Source B. Ref. 2, p. 492

TABLE 3-3

System	Mean Time Between Crashes
1. MIT MULTICS Time Shared System (Under Continuous Development)	3-9 hours
2. Honeywell MULTICS (Stable Version of Multics)	Very much greater
3. A data acquisition program (see Ref. 7)	30
4. A commercial time sharing company running CALL 360 on a 360/50	50
5. A commercial time sharing company running a business information system on a mini- computer with many attached disk files.	500

TABLE 1-2

Hardware and Software Failure Rates (see Ref. 7)

	Failure of	Failures	Failure rate	% of Total
H A R D W A R E	CPU	7	$4.08 \times 10^{-3}$	12.5
	Memory	12	7.00	21.4
	Disk pack	1	.59	1.8
	Fixed head disk	2	1.18	3.6
	Power	3	1.77	5.4
S o f t w a r e	Misc.	4	2.32	7.1
	Operating system	19	11.1	34.0
	Applications prog.	8	4.6	14.0
	Totals	56	$32.8 \times 10^{-3}$	100 %

TABLE 4-1

Computation of Model Constants from the Data of Ref. 8

Program	Size	$E_T/I_T$	$\rho_o$
Supervisory A	210 K	$6.14 \times 10^{-3}$	$0.875 \times 10^{-3}$
Supervisory B	240	7.97	0.996
Supervisory C	230	7.48	1.25
Application A	240	13.20	2.20
Application B	240	7.70	1.54
Application C	240	7.00	1.00
Application D	240	12.90	0.995
Average.....		8.92	1.26

TABLE 1-3

Distribution of Effort by Programming  
Phase for Various Projects (see Ref. 6)

	Analysis and Design	Coding and Auditing	Checkout and Test
SAGE	39%	14%	47%
NTDS	30	20	50
GEMINI	36	17	47
SATURN V	32	24	44
OS/360	33	17	50
TRW Survey	46	20	34

TABLE 4-2

Computation of Model Constants from the Data of Ref. 11

Program	Size	$E_T/I_T$	$\rho_o$
MA	4.03 K	$25.4 \times 10^{-3}$	$2.54 \times 10^{-3}$
MB	1.32	13.7	1.37
MC	5.45	17.1	1.71
MD	1.67	15.6	1.56
ME	2.05	34.6	3.46
MF	2.51	14.7	1.47
MT	2.10	12.4	1.24
MG	0.70	22.9	2.29
MH	3.79	13.2	1.32
MX	3.41	23.4	2.34
Average.....		19.3	1.93



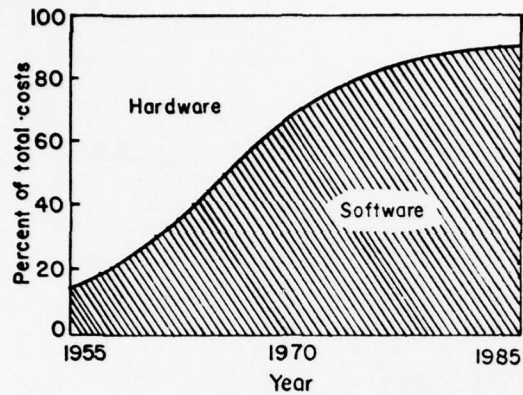


Fig. 1-1. Ratio of hardware to software expenditures in Air Force computers. Data and projections. See Ref. 6.

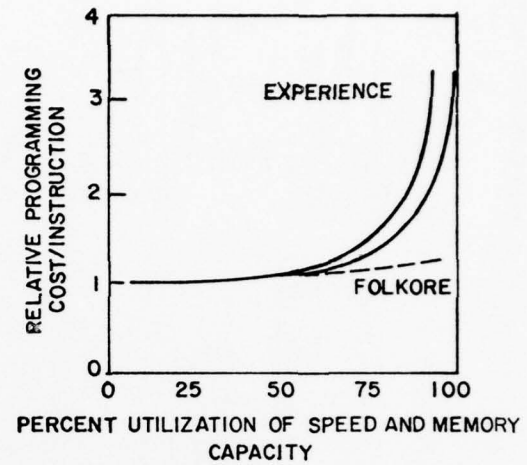


Fig. 1-3. Effect on cost (and reliability?) of high % memory utilization. See Ref. 6.

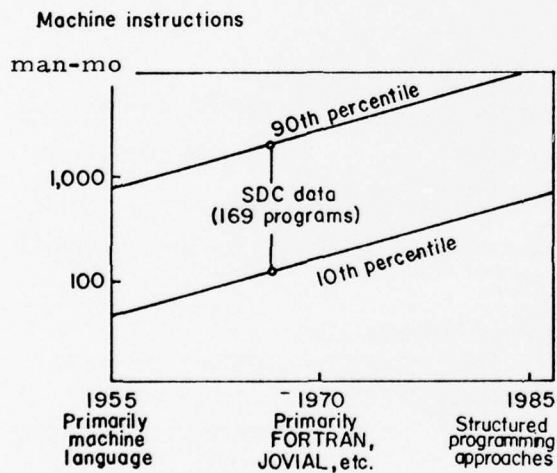


Fig. 1-2. Programming productivity per man month. See Ref. 6.

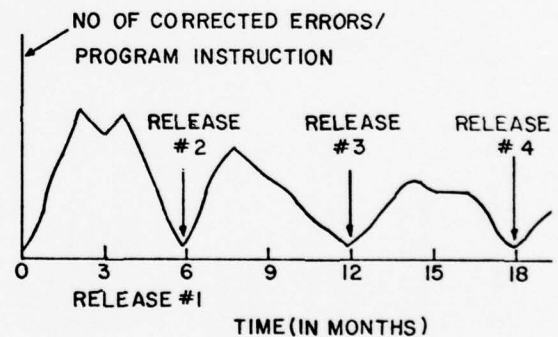


Fig. 4-1. Typical behavior of a compiler program.

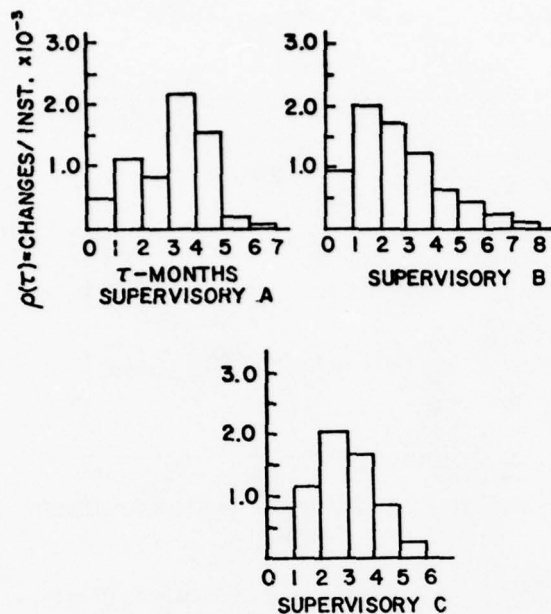


Fig. 4-2. Normalized error rate versus debugging time for three supervisory programs.

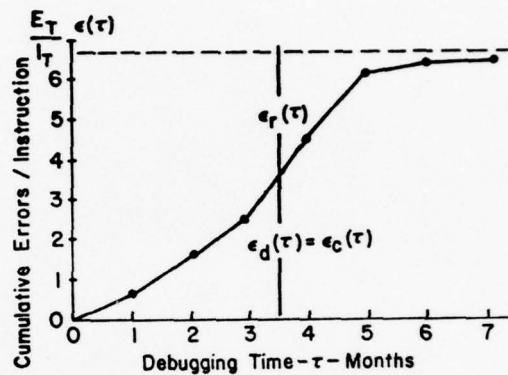


Fig. 4-4. Cumulative error curve for supervisory system A given in Fig. 4-2.

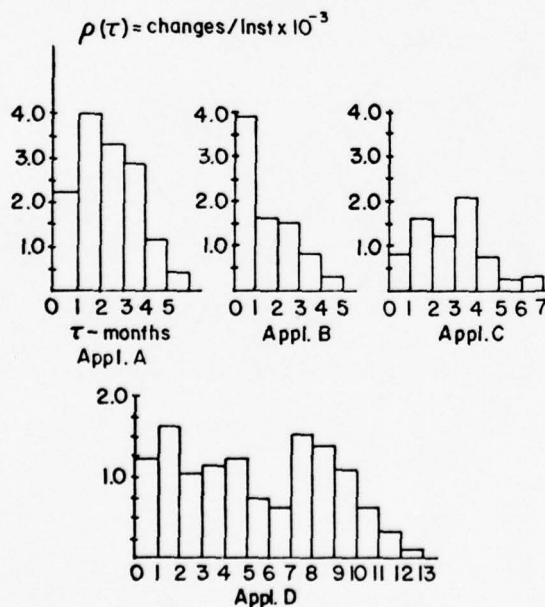
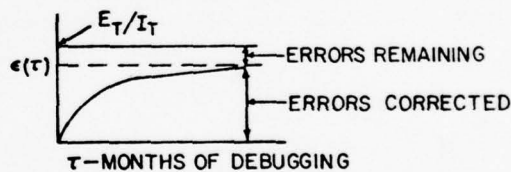
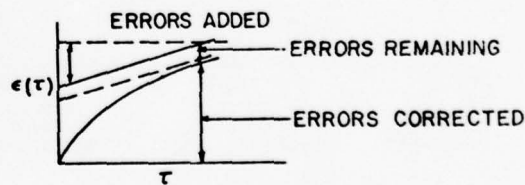


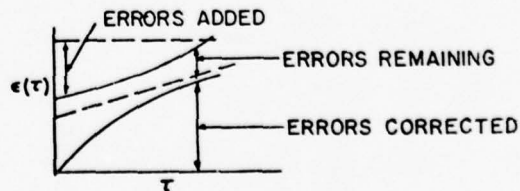
Fig. 4-3. Normalized error rate versus debugging time for four applications programs.



(a) Approaching equilibrium, horizontal asymptote, no generation of new errors.



(b) Approaching equilibrium, generation rate of new errors equals error removal rate.



(c) Diverging process, generation rate of new errors exceeds error removal rate.

Fig. 4-5. Cumulative errors debugged versus months of debugging.

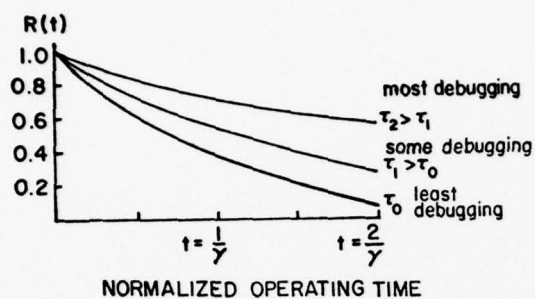


Fig. 5-1. Variation of reliability function  $R(t)$  with debugging time  $\tau$ .

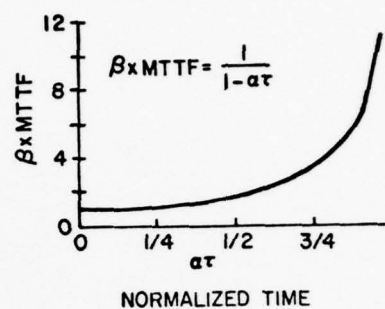


Fig. 5-2. Comparison of MTTF vs. debug time for the constant error debug rate model.

## Software Integrity Through Visibility

G Belcher and T Egan, Flight Controls Division, Marconi-Elliott Avionic Systems Ltd,  
Rochester, Kent ME1 2XX, England

### SUMMARY

This paper discusses visibility in the construction, testing and safety analysis of flight control systems. A computer readable single source software specification which allows information to be retrieved in a more automated manner is used to improve the visibility. A test structure that can display a complete correlation between control functions as specified and as implemented also increases the visibility of the test results. A data flow analysis based on a complete cross reference and symbol table is used as the basis of a method for increasing the visibility of the dormant and common mode failure analysis.

### INTRODUCTION

Digital flight control systems which use software to sequence and perform the control functions in real time require that this software be constructed, tested and analysed with at least the same integrity as the hardware. A major task in the development of this software is the production of sufficient information about the software from the constructing, testing and analysis to satisfy high integrity requirements. The software is defined by an ordered set of alphanumeric characters. This set of characters is used in various ways to produce an equally ordered set of numbers which are loaded into the flight control computer so that the correct sequence of flight control functions is performed.

The process for producing the software for flight control applications must be configured and structured in such a manner that no information is lost or distorted whilst performing the transformation. In order to show that the information content of the software has not been distorted or lost in any way, the software production processes must display what is happening to the set of characters as they move through the various stages of change. As each process is completed, evidence must be presented to show that the functional and safety requirements are still satisfied. It is the visibility provided by the presentation of this evidence that this paper will discuss. It is to be noted that the visibility referred to here means not only formal documentation but more importantly methods for proving the correctness and safety of the flight control software.

The presentation of information will be through line printer listings, visual display units and graph plotters. This note identifies the information required in order to show that the flight control software has preserved its integrity and specifies the visibility requirements for this information when it is displayed.

For the purposes of this note the software is considered to be constructed using structured programming techniques with modules and groups of modules linked by a data module into segments. The module groupings or segments are required so that the various control functions may be called by the executive module at differing iteration rates, e.g Pitch Control Law functions at 50Hz, gain schedules at 10Hz, and structural filters at 100Hz. The complete digital flight control system is considered to be multichannel with similar hardware redundancy and similar software in each channel with a channel identification flag providing, where necessary, run-time software dissimilarity. The hardware/software complexity is such that there are some 200 uniquely identifiable modules, averaging 50 program store locations, each with approximately 5 inputs. The assembler symbol table for this software is required to hold 3000 six character data and program identifiers which are sorted according to symbol name and symbol value and stored on separate files. Symbol tapes are specified, by the assembler as necessary, for use with the software cross-reference table. The software is produced using a macro expand facility, which allows control functions to be uniquely defined and then used, where specified throughout the system. The maintenance of the high integrity, required for this type of flight control software, through visibility during the construction, testing and safety analysis processes applied as the software passes through the software production facility will now be discussed in more detail.

#### 1. SOFTWARE CONSTRUCTION

##### 1.1 Documentation

The software documentation is the medium used for controlling the progress of each module. Software design requirements documents are needed at the beginning of a project so that the software specification and structure can be identified. Once the specification and structure of the software have been defined, the specification and structure documents are used to identify and form the data structure (data module) that will be used by the sequence controller (executive module) to call the various program segments, as and when required, to perform the various control functions. The individual modules will also refer to the above documents for their data and functional structures. Apart from any system or block diagrams, these documents are recorded in the form of alphanumeric character strings. It is these documents that are to be used by the coder to produce,



with the help of an assembler, the numbers that are to be loaded into the aircraft computers. A record of these numbers (code) produced will also form part of the constructional documentation for the software.

In order to preserve the integrity of the flight control software it is essential that this documentation be produced at the beginning of the software assembly and be dynamic enough to be able to control the production of the modules.

The visibility requirement for the documentation that will control the software module through its constructional processes may be met using documents which can be formatted to be computer readable. The formatting required is not of a complicated nature, it consists of various tables and operating system commands.

In this way the total information content of the software specification and structure is stored in a disciplined and controllable manner. The process of making the documents computer readable will in itself improve not only the visibility but also the integrity of the final software produced.

### 1.2 Configuration Control

The major functions of the configuration control system are the control of a) the information as specified, b) the information as implemented, c) the changes to a) and/or b), and d) the project schedule.

The specification information concerning the flight control software is parameters such as module/segment identification, input/output lists, size/runtime, and functional running order. The implementation information is as above but with the added parameters concerning the coding, testing, integration and validation of the modules/segments. The change information is used to control all interaction effects on the other modules/segments.

For project management purposes the visibility requirement is for a dynamic display of the correlation of budgetted with actual progress so that the differences are easily found. The display is usually in the form of build status reports. The use of an automated data base system gives up to date details of budget/actual storage/runtimes, module status, work in progress/outstanding, change status and scheduling information. This information is obtained by using the host computer to continually search for differences between the flight control functions as specified and as implemented. The configuration control system is so structured that when a change, either to the specification or to the implementation information, is requested then a search of the complete data base is possible. This search will itemise the total effect of the change on the flight control software, and will reschedule the work accordingly. Up to date change status is given with its corresponding impact on the software construction schedule. Any potential overruns of storage and runtime requirements are noted in the early stages of the project. Hence, the loss of integrity, which might occur due to the deletion of control functions to conserve the resources of the flight control system will be avoided.

It is to be noted that full character sum checks are required in the implementation of the automation of the configuration control facility. Integrity based on character files which are to be edited and continually updated means that the system must be protected at all stages against loss of characters which will distort the information content of the flight control software specification and structure.

A further improvement in the integrity can be obtained by using the flight control software configuration control procedures on the software required to implement these procedures.

### 1.3 High Level Languages

A major task when specifying and structuring a flight control function or sequence of functions is to give the full visibility required so that the specification and structure can be analysed in order to validate that it is safe. By using a high level language at the specification and structuring stages of a project, an improvement of the visibility in and the integrity of the flight control software constructional processes can be made. The functions required for flight control are essentially simple ones, which are ordered in a complex but uncomplicated manner. The high level language statements will therefore form a very simple subset of those available to that language. A standard high level language is preferred in order to make use of the integrity and visibility already available from its own controls. The high level language used to specify and structure the flight control software can also be used by the configuration control software, and, when required to be compiled, can use the same host computer compiler. Commonality of compilers and languages for a project will, in themselves, improve the integrity of any software produced by that project.

High level languages, by their intrinsic clarity and definition improve the visibility of the software specification. Should the statements also be written with the use of GOTO directives kept to a controlled minimum then the visibility is again improved as far as a validation procedure is concerned. A flight control software function specified by simple high level language statements, backed up with a scaled system diagram, has higher visibility and integrity than a flow chart used for the same purpose. The discipline required to write the high level language statements and the potential they have for host computer processing make them suitable as the media for flight control software specification and structuring. Flow charts, where essential to provide information when

validating the flight control software, are to be added as processing notes only. The improved visibility at the software specification level has also the potential for an improved translation by the coder into the numbers which are to be loaded into the aircraft computers.

Various groups of engineers, with different backgrounds, are required to perform certain tests and validation on the integrated software/hardware flight control system. The use of a high level language as the software specification gives the project a single source for all the information regarding the flight control software. This single source will allow the various groups to work through a common language while they, as individual groups, approach the testing and analysis from different directions. This commonality of language will make the results of the various groups' analysis more visible. This single source is also used for configuration control to further correlate the information as specified with the information as implemented.

#### 1.4 Proof of Correct Software Construction

As the software specification moves through the construction stages and is gradually changed into the numbers that are loaded into the aircraft computer, certain information is presented to the configuration control group so that the "static" checks, as discussed in the previous sections, can be done. These checks take the form of a continual correlation between the specification and its implementation with any differences highlighted. The most rigorous method is to take the numbers from the aircraft computer and process them so that the original specification is produced. The same correlation is performed along the backward path as is done on the forward path. For most of the forward constructional processes, with their associated correlations, there are inverses so that the equivalent reverse constructional processes, with their own associated correlation, can be performed. By structuring the software and its production processes so that these inverses are available, it is possible to provide further proof that the software, as specified, has been implemented correctly. In addition to showing that the software is constructed correctly (partially by performing these "static" checks) the specification and the structure itself must be tested to prove its correctness. This form of testing takes place at execution time, and is discussed in the following sections.

### 2. SOFTWARE TESTING

#### 2.1 Test Documentation

The testing that will be discussed in this section refers to those tests that are to be performed on the flight control software before it is loaded into the aircraft computers. The documentation controlling the testing of the software, as that which controls the construction of the software, should provide the maximum visibility at each stage in the test procedure. Improved visibility and integrity can be achieved by making the test specification and structure documents computer readable so that correlation between expected and obtained results is facilitated. The tests that are performed on the software after the construction and before the loading onto the aircraft computers, are in general, dynamic tests. To improve visibility the individual tests are of a simple nature and are performed using the binary numbers that are to be loaded onto the aircraft computers. These same tests can be run on the high-level language specification. In this way, the results from the specification and the binary numbers representing that specification can be correlated and any differences highlighted by the host computer.

Configuration control is applied to the testing as well as the construction of the software. A major feature of the automated construction of the software is the data module or symbol table. This file records the numerical correspondence for every label or data identifier required by the flight control software and can be used to improve the visibility, not only of the software construction, but also of the software testing. The symbol table file is available when defining the test structure, so that all test specifications can identify the input and output values by their software names, instead of only their numerical or address representation. This improves the visibility in the test results checking procedure and hence helps to preserve the integrity of the flight control software.

Certain integration tests have to be performed on the software modules/segments before any execution tests can be done to check their system performance. The integration tests provide, for validation, such information as input/output lists, module/segment size/maximum runtime and module/segment identification. This information is correlated with the corresponding information as specified in the structure document and the differences, if any, are displayed. The feedback for this cross reference process is supplied by the symbol table. This cross referencing information, usually presented in the form of a cross reference table, is often provided by the assembler program. For improved visibility this cross reference table can be produced from the binary file that is loaded into the aircraft computers. Thus it will be a feature of the software simulator, as well as the assembler. This cross reference table is available for the safety analysis that is to be performed on the flight control software structure.

#### 2.2 Performance Tests

The following limited set of examples illustrate the visibility that can be obtained when testing software on a host computer prior to the loading of the software modules/segments into the aircraft computers.

#### i) Overflow tests

All combinations of maximum and minimum scaled values of the inputs are used to test the module/segment for overflow. All modules followed by a limit function have their outputs correlated with that of the limiting function. It is a requirement that the test structure be able to do this correlation and present the information in the form of a comparison between the module outputs and the specified limiting function. Should the limiting function not be present in the code then the test structure is to highlight this as it formats the above information. For improved visibility and integrity the complete module/limit function information must be available when testing the module for overflow. Similar information is also required for signal selector and monitor threshold tests.

#### ii) Decision path tests

Should a module contain more than one process path then the test structure is to be able to show, by a trace feature, that all the paths are not only there but are capable of being followed. The number of paths must correspond exactly to the number specified. Any discrepancy is to be highlighted. For increased visibility the decisions within a module are to be in the form of a "true/false" structure. When the decision is true, a certain function is performed, when the decision is false, this function is bypassed.

It is suggested that only forward pointing GOTO directives are used to improve the storage and runtime required by the module. This will preserve the simple and hence visible structure of the decision without adversely affecting the complete flight control structure by using too much program store and iteration period.

#### iii) Self contained function tests

When testing the function of a module, improved visibility can be obtained by structuring the flight control software such that each module only performs a single simple function. This function should be able to be performed and tested without using any other function or module available in the software. This avoids the need to retest a module every time another module is changed. For improved visibility the inputs to a module should be available at the time the module is entered, and should not be required to be calculated during the execution of that module. In this way, the values for the inputs can be examined before and after running the module and can, therefore, be shown to have been calculated independently of any other software function. This type of software structure also improves the visibility from the safety and data flow analysis viewpoint because the only way that a module can affect itself and other modules is via its data outputs.

#### iv) Macro and subroutine call tests

Macro expansions can be used with advantage for flight control software to call standard control functions and to load subroutine parameters. These standard functions can be tested separately from the modules that call them, while the calling module has to be tested for the correct positioning of the macro call and for the correct number and naming of any parameters. The use of macros also increase the visibility of the software specification by giving a one to one correspondence between the name of the macro and the function that it performs. The running order for the control functions is also made more visible and can be checked against the system diagram with increased integrity.

The macro as specified can be correlated with the macro as implemented in the symbolic assembler language. The number of actual parameters, and the running order can be compared with those as specified and differences displayed. In order to check that the macro as specified is on the binary file a substitution module is used. This module is designed with reference to the specification for the macro and not with reference to the calling module. The substitution module will improve the visibility of the macro as called by printing, at simulation time, the contents of each parameter, together with the macro identification and position in the running order.

To improve the visibility no macro definition should contain a call to a subroutine and all subroutine parameters should be loaded before and off loaded after the subroutine call. The software specification and the binary file are both checked for correct use of subroutines. Further, all calls to subroutines in the flight control software are documented and labelled so that they can be easily identified during the safety analysis.

### 2.3 Random Input Testing

The tests described above are structured so that the visibility of the software constructional process is improved. This test structure presents ample evidence that the binary file is a correct representation of the software as specified.

The prime purpose of the tests on the host computer is to present evidence, in the most visible form, that the binary file will execute the flight control functions as specified. When the binary file is tested using input numbers rather than by examination of the code, the numbers are to be a simple set, chosen to confirm that the code is correctly representing the software specification at execution time. It is only when this correct representation at execution time has been confirmed that the modules/segments can be released for further testing.



When testing on the host computer, visibility is preserved by limiting the tests to aspects such as simple dynamic and end-to-end resolution tests. More complex tests such as random inputs, are best performed on a hybrid computer or on a system rig since over complex tests on a host computer may decrease the visibility rather than increase it. Thus when testing the flight control software on the host computer, particular combinations of inputs are preferred to random inputs so that the test results can be presented in a simple and easily understood manner. All tests are to be structured so that their results can be used, without ambiguity, for the safety analysis to be performed on the flight control software.

### 3. SAFETY ANALYSIS

#### 3.1 Configuration Control Examination

The Failure Modes Error Analysis (FMEA) for the flight control software must examine the software preparation procedures for the type of errors that can occur and the methods used to remove these errors. The test procedure must be examined to determine the type of errors detected by the various tests. The sources of the test data sets must be examined for possible errors and the effect of these errors on the whole test structure must be analysed. The simple dynamic tests performed on a complete control law function must be checked to see that they exercise all the paths through the function. If any paths are found to be untested at this stage then the function must be retested to exercise them. The authorship of the test specifications must be checked to ensure that the same programmer did not design a module/segment and test the same module/segment. The assembler errors produced while integrating the modules must be examined for weaknesses in the preparation procedures.

The Hardware/Software test procedure must be examined to show that the tests that were too complicated (and hence not sufficiently visible) to perform on the host computer have been included in this type of testing. The configuration control procedures for implementing software change requests and for reworking modules must be examined to ensure that the correct issue of the module is incorporated into the software correctly. The procedures for correcting and redesigning the function of a module must be examined to make sure that the code is completely retested before being integrated with the flight software. The complete test plan for the flight control software must be examined to show that the objective of reducing the untested code to zero has been achieved.

The history of the design and coding errors must be examined in order to establish that the change procedure gives the correct feedback and that weaknesses in the testing methods discovered are corrected and the modules affected are retested. The analysis of the test procedures must show that each test will be successful in uncovering the type of error for which it was designed. The software specification must be reviewed as the module test results become available. This review will require a FMEA to be done on the module at the time that it is tested. In order to improve the visibility and integrity of the complete software FMEA it is recommended that each type of test has its own associated failure modes and error analysis, which is to be done at the time the test is performed.

#### 3.2 Cross Reference and Data Flow Analysis

The cross reference information provided by the software simulator is obtained from the binary file. This must be presented in a simple visible manner, e.g. complete lists of inputs and outputs, lists of modules giving their inputs and outputs, lists giving the position of every subroutine call. These lists will also give the same information as specified, with the differences highlighted. A more visual way for representing this cross reference information could be in the form of a software "wiring" diagram similar to that used for the hardware. In addition to this data connection display, the timing information particular to a module must be presented in a visual form. Data flow analysis is an essential feature of the software FMEA. All modules must be shown to be in the correct time "slot" for the flight control functions. The problem with data flow analysis concerns the presentation of the information in the most visible form so that the analysis can be done with high integrity. It is suggested that an interactive method, involving the host computer, is a suitable method of improving the visibility of the data flow analysis. The system diagram as specified is to be used, together with the cross reference data, during the interactive process to enable the operator to follow any data path from sensor input to system output command. The interaction is required so that the modules of interest can be quickly and efficiently specified and checked by the operator. Hard copy is produced so that the flow for any particular data item can be analysed. Any timing errors or data skew are highlighted on the printout.

#### 3.3 Dormant Failure Analysis

One of the advantages in designing the flight control software in the form of structured modules each performing a simple function is that the dormant failure analysis can be conducted by considering that the malfunction of a module is due to the corruption of the input data by a prior module. The module under consideration is assumed to be working correctly. Thus a dormant failure is any fault that causes an erroneous input to be passed to a module such that any errors in both the input and output remain undetected.

The type of dormant failure that can cause one or more good channels to be in error is of particular concern for the analysis. To increase confidence, the data flow analysis can be used to follow the path of data from any particular input and check what its



effect is and whether it will be monitored. The results of this analysis are presented in such a manner that it can be shown that all inputs and all data paths have been considered. Failure cases that are still suspect after this analysis are examined further, with in some cases, simulation of the failed cases on the host computer.

The software specification for the monitor structure is also examined for particular dormant failures. Each module containing any monitoring function is examined for erroneous inputs and outputs. Again, simulation on the host computer will improve the visibility of the results.

### 3.4 Analysis of Discontinuities

As part of the complete common failure mode analysis, the arithmetic as defined in the software specification is studied for possible discontinuities. Typically the code is examined for instructions that can cause an overflow condition to occur. This search can be automated and the information can be displayed with improved visibility by means of the assembler symbol table. Typical information printed will give the program location at which the overflow can occur, together with the variable names that are able to contribute to this overflow. Possible effects of the overflow are also to be analysed, by means of the data flow analysis.

After identifying all the program positions at which overflow can occur, the range of the numbers contained in the contributing variables is examined to identify what particular range of input values will cause the overflow to occur. The data flow analysis will also be used to follow these particular module inputs back to the sensor inputs.

### CONCLUSION

Techniques have been developed which give good visibility of flight control software during the writing of the specification, the testing on the host machine and in the analysis in the corresponding FMEA. These techniques combined with programming disciplines during the construction phases and dissimilarly in the subsequent testing of the integrated hardware/software give a high confidence that the similar software will be free from common mode errors.

### REFERENCES

1. J. Maynard, "Modular Programming", Butterworth (Publishers), London, 1972.
2. J. C. Hall and D. L. Hemmel, (Collins Avionics Division, Rockwell International) "An Approach To Efficient Design" - Arinc Avionics Engineering Seminar On DAFCS, Washington, D. C., May 1975.
3. E. F. Miller and M. R. Paige, "Automatic Generation of Software Test Cases", Euro Comp 1974 Proc.
4. Ministry of Defense (UK), "Specifications For Air Technical Publications", AVP 70, Specification 4, November 1973.
5. Naval Ordnance System Command, "Requirements For Digital Computer Program Documentation", (WS-8406), Revision 1, November 1971.
6. R. Ruggles, (Marconi-Elliott Avionics Systems Limited) "Computer Architecture and Technology", Presentation material given at AGARD Consultant Mission, London, May 1976.
7. J. E. Templeman, (Boeing Commercial Airplane Company), "Software Considerations" Presentation material given at AGARD Consultant Mission, London, May 1976.

## TECHNIQUES FOR MICROPROGRAM VALIDATION

W. C. Carter, H. A. Ellozy, W. H. Joyner, Jr., G. B. Leeman, Jr.

IBM Thomas J. Watson Research Center, Yorktown Heights, USA

SUMMARY

A method for computer-assisted verification of systems controlled by microcode will be presented, with examples of its application to actual implementations. Such systems may be a computer with operations emulated by microcode action on a simple processor or a microprocessor coded to perform a fixed task. The specifications for such a design and those for the processor on which it is to be implemented are both described formally, with the code to be certified supplied as data to the low-level description. Informally, correctness of the implementation means that if the specification description and the machine description begin computation with identical inputs, they will get identical results. An interactive system of programs for carrying out mathematical proofs of a formalization of this correspondence has been written. Its application-independent monitor provides a framework for a goal-directed attack on the problem, allowing the user to reduce it to subproblems; programs are invoked to perform symbolic interpretation of the descriptions, generation of sufficient conditions for correctness, theorem proving, and simplification. This system is running and has been used to detect and correct errors in an actual microcode implementation; preliminary results of these experiments are described.

LIST OF SYMBOLS

LSS	Language for Symbolic Simulation
SVAE	Symbolic Validation of Algorithmic Equivalence
APL	A Programming Language (name)
MCS	Microprogram Certification System
BNF	Backus Naur Form
LISP	Programming Language implemented for many computers
System/360	IBM Computer Family
macro	subroutine
expnf	expnf is a LSS macro name
L	Leaf of a control tree
F <sub>j</sub>	jth Function
D <sub>k</sub>	kth Domain
R <sub>j</sub>	jth Relation
P	Program (abstract)
$\delta$	algorithmic mapping in LSS semantics
$\rho$	mapping in LSS semantics
S	abstract state
S Machine	Hypothetical Computer, well described
MEM	Memory (Storage)
STK	Stack Pointer, S Machine
SX	Index Register, S Machine
SCC	Instruction Counter, S Machine
SSW	Stop-Start Switch, S Machine
s-control	control tree in LSS semantics
MAR	Memory Address Register
$\mu$ S	First Micromachine Data Flow
MDR	Memory Data Register
X	Index Register
A,B	General Purpose Registers
CC	Instruction Counter
ALU	Arithmetic and Logic Unit
LATCH	Register used for circuit timing
IR	Instruction Register
E,H	Logic Signals, in $\mu$ S Diagram only
CONTROL	Storage for Microprogram, $\mu$ S Data Flow
$\mu$ CODE2	Microcode which emulates S on $\mu$ S
IFETCH	machine cycles which determine instruction operands
IEXECUTE	machine cycles which execute instruction
I/O	Input/Output
n K	n thousand units
$\mu$ S'	2nd Micromachine data flow
D	Decision Register
EMITS	Registers which get data directly from microinstructions
STATS	Flip Flops
Y	output (5 bits) from STATS
ns	nanosecond
$\mu$ P1	Microcode which emulates S on $\mu$ S' (no temporal overlap)
$\mu$ P2	Microcode which emulates S on $\mu$ S' (with temporal overlap)
IROM	Instruction Read Only Memory
CTR	Counter
SIMULATE	Goal class in MCS
TRACEIT	Goal class in MCS
THMPRV	Goal class in MCS

RSPLIT	Rule in MCS
TRACEM	Rule; symbolically execute algorithmic level
TRACEMUM	Rule; symbolically execute implementation level
PROVEIT	Rule; invoke the theorem prover
GENVC	Rule; generate verification conditions
ANDSPLIT	Rule; in theorem prover
CASESPLIT	Rule in MCS
\$ (Symbol)	Symbolic value assigned to symbol
MSLIB	Macro library
exec-pgm	macro, execute program, in S macro library
exec-mpgm	macro, execute microprogram, in $\mu$ S macro library
QA4	Theorem Proving System
SCRATCHPAD	Name of symbolic algebraic programming system
HTC	Hybrid Technology Computer
SUMC	Space Ultra-reliable Modular Computer
TSE	Test Support Equipment
IPL	Initial Program Load

#### LSS APL-LIKE OPERATORS

,	catenation, joins two vectors
[ ]	index, b[3,5] gives the 3rd and 5th elements in b, a vector
[;]	array (two dimensional) indexing A[n;]nth row of A
i	index generator, i5 first 5 integers, 0,1,2,3,4
$\rho$ A	shape (dimension) of A
V $\rho$ A	reshape A to dimension V
+	plus (over integers)
a+2b	sum of constant and index generator: 2+23 = 2,3,4
-	minus
=	equality (in normal mathematical sense)
$\leq$	less than or equal to
$\wedge$	logical AND
$\vee$	logical OR
$\sim$	logical NOT
?	Random choice
Via	Decode V to base a
*	exponentiation
Vta	Encode V in base a
:	assign
$\leftarrow$	replacement
$\rightarrow$	implication
$\Omega$	empty leaf

#### BNF OPERATORS

: =	is defined as
	or
( )	recursive definitions are used parenthesis

### I. INTRODUCTION

A method for computer-assisted verification of systems controlled by microcode will be presented, with examples of its application to actual system implementations. Such a system is a computer with operations emulated by microcode action on a simple processor or a microprocessor coded to perform a fixed task.

The increased use of microcode, residing in a control store or in a main memory, for implementation of the control for such systems makes a demonstration of the correctness of the code a necessary part of the design verification of the system. Indeed, such microcode may be more prone to subtle and obscure errors, which are more difficult to detect using test cases, than programs in a higher-level language [23]. But while several approaches have been developed for proving programs correct (see [7, 12, 16, 31]), little attention has been given to the verification of low-level code. Below we describe a partially automated system which has been used to detect errors in microcode and to certify microprograms as correct.

In proving the correctness of microprogrammed implementations, all of the facets of machine operation must be explicitly described. Correctness of the description and a proof of the correspondence of the assembled code to a given specification guarantee correct execution. However, particularly for microcode implementations of high level architectures, assertions for correctness such as those due to Floyd [10] are not easily formulated. Our approach is to give the specifications for correct implementation as an abstract machine schema [3,20] having a well specified tree control structure which operates upon a state vector of machine components and is determined by a library of macro routines. The state vector components and the macros are written in the Language for Symbolic Simulation (LSS) whose syntax and semantics will be described subsequently. The attributes of the computer on which the specified architecture is to be implemented are also embodied in such an abstract machine, and the desired relationships between the two machines are specified and then established.

The type of equivalence we wish to hold in order to prove correctness has been formalized by Milner as algebraic simulation between programs [25]. As related to our abstract machine descriptions, this notion, Symbolic Validation of Algorithmic Equivalence (SVAE), gives a mechanism for describing both the

points of control in the two machines at which a relationship between them should hold, and at each of these points the desired relationship between the components of the state vectors. A proof that such a relation is a simulation relation, in the sense of Milner, is then a proof that whenever, and however, control in the two abstract machines goes from one pair of corresponding points to another pair, these relationships will hold. The theorems validating this approach will be proved.

Previously, simple examples have been done using SVAE [3, 19, 20] to prove the correctness of microcode. However, these proofs, though formal, were carried out by hand. They will be reviewed briefly so that the basic ideas of algebraic simulation are made clear. For each pair of stopping points corresponding to a simulation component we must: (1) assert that the simulation conditions corresponding to that component hold, instantiate the most general values such that these conditions hold in the state vector, and perform immediate simplification; (2) run each abstract machine, performing symbolic computation until another stopping point is reached; (3) verify that the pair of points reached corresponds to a component of the simulation relation; and (4) prove that the simulation conditions of this component hold. It became apparent from this work that the individual parts of such proofs were not of great complexity, and that the main impediment to human proofs was the generation and organization of these many separate parts. Since this number of parts increases with the size of the implementation being verified, it became clear that some automated aid would be necessary.

An interactive system of programs for carrying out mathematical proofs of a formalization of this simulation has been written. Its application independent monitor provides a framework for a goal-directed attack on the problem, allowing the user to reduce it to subproblems; programs are invoked to perform symbolic interpretation of the descriptions, generation of sufficient conditions for correctness, theorem proving, and simplification. This system, the Microprogram Certification System (MCS) will be briefly described. It is running and has been used to detect and correct errors in an actual microcode implementation; preliminary results of these experiments are described.

## II. LANGUAGE FOR SYMBOLIC SIMULATION

The first step in the SVAE method is to define formally and precisely the algorithm and its implementation. In any discussion of computer design, the natural levels of system description must be carefully defined. The architectural level contains the attributes of a computer system as seen by a programmer; i.e. the conceptual structure and functional behavior as distinct from the organization of data flow and controls, the logical design, and the physical implementation [2]. Microprogramming was first proposed as a technique for the orderly design of logic to control the processor data flow. This level is frequently called the register-transfer level. For several systems of the 1960's, such as the IBM System/360, simulators were devised which accepted as input computer descriptions at the register-transfer level, and instructions for standard assembly level program debugging techniques [5]. However, as the use of microcode for emulation, diagnostics and special functions increased, this technique of microprogram certification became less viable. To validate a computer design using microcode, we first define formally and precisely the computer at both the architectural and register-transfer levels. The operation modeled is the successive execution of strings of instructions, beginning with the START button being pushed and ending with either a STOP or an ERROR. As discussed in [18], the necessary primitives for such a modeling system are: (1) a data set, (2) a closed set of operations over the data set, and (3) a set of operations to control the order in which operations on the data set are performed. The data set and operations should be chosen so that their application in specific instances will result in implementations which can be easily manipulated to produce theorems whose satisfaction will prove the validity of the given design. These manipulations and theorems should lend themselves to mechanical treatment via computer assistance. The language in which we shall describe the formal models which allow validation to be proven is the Language for Symbolic Simulation (LSS). The syntax of LSS will allow the data set and the set of operators to be described. The LSS semantics will completely describe the order in which operations on the data set are performed.

The description of the syntax begins with a description of a facility vector, following [24]. The facility vector is, for our present purposes, a list of the components of the machine: registers, storage switches, lines, etc. We associate with each of these a shape, or dimension, in the style of APL. Registers have associated with them their width in bits; main storage has the dimensions of a matrix. The operations are defined by a macro library. The macro library for each description is a list of macro definitions for an abstract machine with a tree control structure. Each definition consists of its name, formal parameter list, and either a tree into which the macro expands, or several such trees whose selection depends on predicates over elements of the facility vector. These trees contain other macro calls and assignments of values of APL expressions [9] or values returned by macros to local variables or to elements of the abstract syntax. The formal description of the LSS syntax in BNF follows:

```

facility-vector := (varb-desc-1...varb-desc-n)
varb-desc      := (id shape) | (id shape permanent-value)
permanent-value := APL-expression (over constants)
shape          := ordinal | (ordinal-1...ordinal-n)
term           := facility-variable | local-variable | array-reference | PASS
arg            := APL-expression
exp            := arg | macro-call
macro-call     := (macro-name arg-1,...,arg-n)
assignment     := (term : exp)
item           := macro-call | assignment
tree           := (root) | (root tree-1...tree-n)
root           := item
pred-expan     := (predicate tree) | (predicate pred-expan-1...pred-expan-n) {n>1}
macro-def      := (macro-name param-list tree) | (macro-name param-list pred-expan-1...pred-expan-n) {n>1}
param-list     := (id-1...id-n)
macro-library  := (macro-def-1...macro-def-n)

```



Figure 1 shows the facility vector for the architectural description of the S-machine, described in Section V.

```
{(MEM    (16777216 32)
  (STK    32)
  (SX     32)
  (SCC    32)
  (SSW    1)}
```

Example of facility vector showing dimension of each machine component

Figure 1

Figure 2 is an example of a macro definition.

```
execpgm =
SSW = 1 → execpgm
          execinstr (op, adl)
          advctr
          adl: instrprep (id, ix, op, ad)
              (id: a[0]; ix: a[1]; op: a[2+26]; ad: a[8+224])
              a: fetchword (SCC)

SSW = 0 → Ω
```

A tree control macro

Figure 2

Following Landin [17] and McCarthy [24] the LSS semantics will be defined by the operation of an abstract machine.

M: (S, I, S<sub>i</sub>, S<sub>f</sub>, μ, ρ, G)

where S is the set of all states  
 I is the set of input states  
 S<sub>i</sub> is a specified set of initial states  
 S<sub>f</sub> is a specified set of final states  
 ρ: S → S is the transition relation mapping  
 μ: I → S is a mapping from I to S<sub>i</sub>  
 G is a goal function map from S<sub>i</sub> × S<sub>f</sub> to {0,1}, i.e. the desired action of M is known.

The mapping ρ maps states into states nondeterministically as defined by 1) a macro library in LSS syntax and ii) an algorithm S which carries out the control steps necessary to interpret the LSS macros so that the proper operations are carried out on the states. The algorithm S will be discussed later. The set of states S = {S<sub>j</sub>}, where S<sub>j</sub> is the catenation of the LSS facility vector and s-control, a control tree with LSS item at each mode.

The mapping S is defined briefly as follows. If the leaf L of the control tree is a macro, then L is replaced by the expansion of the macro if the latter contains no predicates. If the macro has one or more nested paths of predicates, the first path with all its predicates true is chosen; the tree structure at its end replaces L. If L is an assignment statement, then either local or global variables are appropriately modified. The detailed step by step algorithmic definition follows.

- 1) If s-control is the null tree Ω, stop.
- 2) Choose any leaf L from s-control, and let L' be the leaf immediately above L.
- 3) If L has the form v:t, where v is a term and t is either a macro name or an APL expression, then replace L on s-control by v, and add a new leaf t to s-control below L; go to step 1).
- 4) If L has the form PASS: t, where t is as in step 3), replace L by t; go to step 1).
- 5) If L is a macro name, find its expansion m in the macro-library, and perform proper passing of arguments.
- 6) If m has no predicates, replace L by m, performing proper local variable binding; go to Step 1).
- 7) If m has predicates, choose the first predicate which is true, and replace m by the structure to the right of this predicate; go to step 6).
- 8) If L is an APL expression (in particular, a variable name), evaluate it, assign it to L', and remove both L and L' from s-control; go to step 1).

Note that when s-control = Ω, S is in effect the identity function. As an example, suppose s-control has one leaf fetch-word (ic), where ic indicates instruction counter. In the expansion

fetch-word(t) = PASS: mem[m];

m: 2 + t[8 + 224]

t is replaced everywhere by the value of ic, and the binding mechanism insures that value 2 + ic[8 + 224] of the local variable m will in fact be inserted into the PASS statement, yielding PASS: mem[2 + ic[8 + 224]];]. Thus the effect of this sequence is to fetch the current machine instruction from memory.

The mapping  $\mu$  is necessary because real machines physically start and stop (usually by actions such as pushing buttons on the console), interior circuits are reset (again by a button) so that specified, known initial states ( $S_i$ ) are reached and instructions as defined in the program manuals can be executed.

In validating computer design, the architectural or specification level machine is completely defined by formalizing its principles of operation as an abstract machine. The second machine specification results in a general purpose computer controlled by the instructions in the control store. This computer is made into a special purpose machine by associating the actual code to be verified as a value of the component which contains it (e.g. the actual microcode in the control store).

After machine levels have been formally described, in order to validate computer designs in practical cases, abstract simulation will be used, so the method of Symbolic Validation of Algorithmic Equivalence will be described.

### III. SYMBOLIC VALIDATION OF ALGORITHMIC EQUIVALENCE

The type of equivalence we wish to hold in order to prove correctness has been formalized by Miller as algebraic simulation between programs [25]. As related to our abstract machine descriptions, this notion gives a mechanism for describing both the points of control in the two machines at which a relationship between them should hold and at each of these points the desired relationship between the components of the state vectors, (reducing the work over proving state equivalence). A proof that such a relation is a simulation relation, is then a proof that whenever, and however, control in the two abstract machines reaches a pair of corresponding points, these relations will hold.

Assume that the algorithm  $S'$  and its emulator  $\mu S'$  have been formally described in LSS. To show that  $\mu S'$  is a correct implementation of  $S'$  we shall employ two concepts. The first of these is an abstract program.

Definition 3.1: Let  $D$  be the union of mutually disjoint domains  $D_{in}$ ,  $D_{comp}$ , and  $D_{out}$ . An abstract program is a pair  $P = \langle D, F \rangle$  where  $F$  is a function satisfying

- 1)  $F(D_{in} \cup D_{comp}) \subset D_{comp} \cup D_{out}$
- 2)  $F(t) = t$  whenever  $t \in D_{out}$ .

There are infinitely many ways to represent  $S'$  and  $\mu S'$  as respective abstract programs  $\langle D', I' \rangle$  and  $\langle \mu D', \mu I' \rangle$ . For example, we can write  $D' = D_{in}' \cup D_{comp}' \cup D_{out}'$ , where

$$D_{in}' = \{ t \mid is-S' \quad (t) \wedge s\text{-control}(t) = \text{start}(t) \} \quad (1)$$

$$D_{comp}' = \{ t \mid is-S' \quad (t) \wedge (s\text{-control}(t) \neq \text{start}(t) \wedge s\text{-control}(t) \neq \Omega) \} \quad (2)$$

$$D_{out}' = \{ t \mid is-S' \quad (t) \wedge s\text{-control}(t) = \Omega \} \quad (3)$$

Next let  $t \in D_{in}' \cup D_{comp}'$  and operate the interpreter: run the algorithm  $\delta$ , starting at step 1) and stopping when step 1) is again reached; call the transformed abstract machine state  $I'(t)$ . We put  $I'(t) = t$  when  $t \in D_{out}'$ , which is in accordance with the definition of  $t$ . Programs  $\langle \mu D_1', I_1' \rangle$  are defined similarly by replacing  $S'$  with  $\mu S'$  in (1), (2) and (3).

Observe that  $D', I'^n$  is also a realization of  $S'$  as an abstract program,  $n = 1, 2, \dots$ , and in this sense  $\langle D', I' \rangle$  is the "finest" (i.e., most detailed) representation of  $S'$  in which we will be interested. More generally, the integer  $n$  may become a function of the current machine state. For example, if  $S'$  is known to contain in its memory a machine language program which always terminates, the "coarsest" (least detailed) representation of  $S'$  would be  $\langle D', F' \rangle$  with  $F'(t) = I'^n(t)$ , where  $n$  is the smallest integer for which  $I'^n(t) \in D_{out}'$ . We shall standardize this notation.

Definition 3.2: Let  $E$  be a set. Then by

$$F'(t) = I'^n(t) \text{ wrt } E$$

we mean that  $F'(t)$  is defined to be  $I'^n(t)$ , where  $n$  is the smallest positive integer for which  $I'^n(t) \in E$ . (A similar notation will be used for  $\mu F'$ ). The point we wish to stress is that the execution function in an abstract program can be adapted to the particular problem at hand.

The second notion we need is that of simulation between abstract programs.

Definition 3.3: Let  $P = \langle D, F \rangle, P' = \langle D', F' \rangle$  be abstract programs and  $R = R_{in} \cup R_{comp} \cup R_{out}$  be a relation, with  $R_{in} \subset D_{in} \times D_{in}', R_{comp} \subset D_{comp} \times D_{comp}'$ , and  $R_{out} \subset D_{out} \times D_{out}'$ .

- 1)  $R$  is a weak simulation of  $P$  by  $P'$  if  $\langle s, F'(t) \rangle \in R$  whenever  $\langle s, t \rangle \in R$ .
- 2)  $R$  is a strong simulation of  $P$  by  $P'$  if  $R$  is a weak simulation,  $R_{in}$  is total, and  $R_{comp}$  and  $R_{out}$  are one-to-one.

As Milner points out, we have the equivalence

$$\langle F(s), F'(t) \rangle \in R \wedge \langle s, t \rangle \in R \Leftrightarrow R F' \subset F R. \quad (1)$$

This second characterization of weak simulation states that  $R$  is a weak homomorphism between  $P$  and  $P'$ . For additional approaches toward simulation which exploit the categorical structure implicit here, consult Goguen [14] and Burstall [6]. Note that there always exists a nontrivial weak simulation between any two programs which never terminate. In fact, if  $P = \langle D, F \rangle$  and  $P' = \langle D', F' \rangle$  satisfy  $F(d) \in Dcomp \forall d \in D$  and  $F'(d') \in Dcomp' \forall d' \in D'$ , respectively, then it is easy to see  $R = D \times D'$  is a weak simulation of  $P$  by  $P'$ . We can give an intuitive explanation of a strong simulation  $R$  of  $P$  by  $P'$ . Set  $Q = Rcomp \cup Rout$ , and let  $s \in Din$ . Then  $\exists t \in Din'$  with  $\langle s, t \rangle \in R$ ; thus  $\langle F^n(s), F'^n(t) \rangle \in Q$ , whence  $F^n(s) = Q^{-1}F'^n(t), n=1,2,\dots$ . Consequently,  $P'$  can compute anything computed by  $P$ .

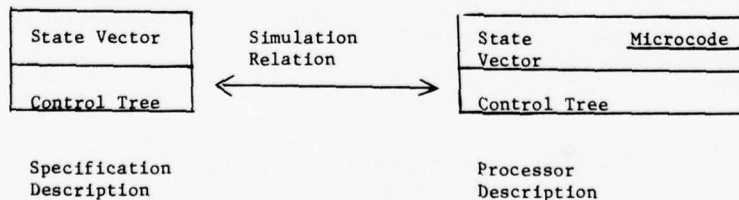
Our definition of strong simulation differs from that of Milner in two ways. First, we need  $Rcomp$  to be both total and one-to-one, whereas Milner imposes no conditions on  $Rcomp$ . In our case it could happen that the memory of  $S'$  contains a nonterminating program of machine instructions, so that  $D', F'$  never leaves  $Dcomp'$ . Nevertheless, we want to show that the micromachine correctly emulates such a program. Thus we need the added assumptions in the computation domain, not merely in the output domain. Second, we relax the requirement that  $Rin$  and  $Rout$  be both total and one-to-one, because our weakened hypotheses are sufficient to establish Milner's Theorem 3.4 (see the parenthetical statement following his proof).

In order to simplify the proofs of simulation by breaking them into parts (see [20]) we represent the simulation relation between abstract machines in a way which differs from the formal notation of Milner. Usually the points of control at which we are interested in establishing a correspondence (the stopping points at which the state vector values determine the intermediate domains which define  $F$ ) are a small subset of all possible values of the LSS control tree. Since parallel operations are described by multiple leaves on control trees [1], the question of the determinacy of  $F$  arises. Our approach thus far has been to show that they can be transformed into linear trees (i.e., that the order of selection of operations is immaterial). See [28, 30].) Therefore (see [8, 21]), we decompose the simulation relation  $R$  into components  $R_1, \dots, R_n$ , one for each pair of control points at which a relation is to be established. Each component contains both control information, specifying for each machine the point of control at which a correspondence must hold, and simulation conditions, detailing the desired correspondence. The control information usually consists of a certain form of control tree, but may also include predicates (stopping conditions) over the state vector variables which further constrain the points of simulation. The simulation conditions or simultaneous verification conditions [21], are in general predicates (usually equalities) relating the state vector variables of the two machines. A sample simulation component appears in Figure 3; Figure 4 illustrates the simulation problem.

<u>exec-pgm</u>	control tree for specification level (single macro on tree)
NIL	stopping conditions (none)
<u>exec-mpgm</u>	control tree for processor transfer level
1 = 21CSAR	stopping condition
MSMEM = SMEM	simulation conditions
MSX = SX	
MSCC = SCC	
MSSTK = SSTK	
MSSW = SSW	

Elements of Simulation Component

Figure 3



Simulation applied to abstract machines

Figure 4

This decomposition of the simulation relation suggests a decomposition of the problem of proving simulation. With the relation partitioned in this way, we must, for each pair of stopping points corresponding to a simulation component: (1) assert that the simulation conditions corresponding to that component hold; (2) run each abstract machine until another stopping point is reached; (3) verify that the pair of points reached corresponds to a component of the simulation relation; and (4) prove that the simulation conditions of this component hold. Below we describe the theorems which show that this procedure is valid.

## IV. THEOREMS ON WEAK SIMULATION

In this section we state and prove two theorems useful for establishing weak simulation. Applications will appear in the next section. Often it is convenient to break up the execution function  $F$  into a composition  $F = F_1 F_2 \dots F_n$  and examine the machine states at each intermediate point. The situation can be depicted by the diagram in Fig. 5 for the abstract programs  $P = \langle D, F \rangle$  and  $P' = \langle D', F' \rangle$ , where  $D = D_0 = D_n$  and  $D' = D'_0 = D'_n$ . As an example, we could take  $n = 2$ , and  $F_1$  and  $F'_1$  could execute IFETCH for  $S'$  and  $\mu S'$ , respectively, while  $F_2$  and

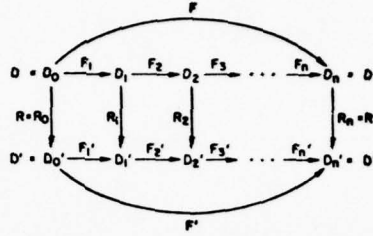


Fig. 5. Diagram for Theorem 4.1

$F_2'$  could perform IEXECUTE. Our first theorem determines under what conditions such a diagram yields a weak simulation.

Theorem 4.1: Let  $R \subset D \times D'$   $\cup D_{\text{comp}} \times D_{\text{comp}}' \cup D_{\text{out}} \times D_{\text{out}}'$  be a relation, and suppose there exists domains

$$D = D_0, D_1, \dots, D_{n-1}, D_n = D$$

$$D' = D'_0, D'_1, \dots, D'_{n-1}, D'_n = D'$$

and functions

$$F_k: D_{k-1} \rightarrow D_k$$

$$F'_k: D'_{k-1} \rightarrow D'_k, \quad k = 1, \dots, n$$

such that

$$F = F_1 F_2 \dots F_n$$

$$F' = F'_1 F'_2 \dots F'_n.$$

1) If there exist relations

$$R_0 = R, R_1, \dots, R_{n-1}, R_n = R$$

such that the diagram in Fig. 5 semicommutates, then  $R$  is a weak simulation of  $P$  by  $P'$ .

2) Conversely, if  $R$  is a weak simulation of  $P$  by  $P'$ , then there exist relations

$$R_0 = R, R_1, \dots, R_{n-1}, R_n = R$$

such that the diagram in Fig. 5 semicommutates.

Proof:

1) To say that the diagram semicommutates means that  $R_{k-1} F_k \subset F_k R_k, k = 1, \dots, n$ . Thus  $R F'_1 = R_0 F'_1 \dots F'_n \subset F'_1 R_1 F'_2 \dots F'_n \subset F'_1 F'_2 R_2 F'_3 \dots F'_n \subset \dots \subset F'_1 F'_2 \dots F'_n R_n = F R$

and we conclude by (1) that  $R$  is a weak simulation of  $P$  by  $P'$ .

2) To prove the second assertion, set  $R'_0 = R$  and  $R'_k = \{ \langle F_k(s), F'_k(t) \rangle \mid \langle s, t \rangle \in R_{k-1} \}, k = 1, \dots, n$ . (2) We claim that  $R'_n \subset R$ . To see this, let  $\langle s_n, t_n \rangle \in R'_n$ ; then  $\exists \langle s_{n-1}, t_{n-1} \rangle \in R'_{n-1}$  such that  $\langle F_n(s_{n-1}), F'_n(t_{n-1}) \rangle = \langle s_n, t_n \rangle$ . Continuing in this way, we obtain  $\langle s_{k-1}, t_{k-1} \rangle \in R'_{k-1}$  with  $\langle F_k(s_{k-1}), F'_k(t_{k-1}) \rangle = \langle s_k, t_k \rangle, k = 1, \dots, n$ . Thus we have

$$\langle s_0, t_0 \rangle \in R'_0 = R, \quad \langle t_0, t_1 \rangle \in F'_1, \dots, \quad \langle t_{n-1}, t_n \rangle \in F'_n$$



whence

$$\langle s_0, t_n \rangle \in RF'_1 \dots F'_n = RF' \subset FR. \quad (3)$$

Now the conditions

$$\langle s_0, s_1 \rangle \in F_1, \dots, \langle s_{n-1}, s_n \rangle \in F_n$$

imply that

$$\langle s_0, s_n \rangle \in F_1 \dots F_n = F, \quad \text{i.e., } F(s_0) = s_n. \quad (4)$$

But (3) says  $\exists u$  with  $\langle s_0, u \rangle \in F$  and  $\langle u, t_n \rangle \in R$ , thus (4) forces  $u = s_n$ ; hence  $\langle s_n, t_n \rangle \in R$ , and  $R'_n \subset R$ , as claimed.

Next,

$$R_{k-1}' F'_k \subset F_k R'_k, \quad k = 1, \dots, n$$

by (1) and (2). Thus the second part of theorem follows if we take

$$R_k = R'_k, \quad R_n = R, \quad k = 0, \dots, n-1.$$

Theorem 4.1 treats the case when  $F$  and  $F'$  can be written as function compositions. However,  $F$  can also be viewed as a set of ordered pairs, and such a set can be broken into mutually disjoint subsets. This situation is illustrated in Fig. 6.



Fig. 6. Diagram for Theorem 4.2

As an example, we mention that sometimes machine instructions can be broken into two classes: the one-address instructions, which calculate an address operand; and the zero-address instructions, which manipulate the stack and do not compute an address. We can take  $n = 2$  and let  $F_1, F'_1$  correspond to the fetching of a zero-address instruction and  $G_1, G'_1$  correspond to its execution.  $F_2$  and  $F'_2$  do the fetching and the address calculation for a one-address instruction, while  $G_2$  and  $G'_2$  perform the remainder of the instruction execution. Our second theorem deals with such a function decomposition.

Theorem 4.2: Let  $F$  and  $F'$  be broken into decompositions as shown in Fig. 5, and let  $R \subset D_{in} \times D_{in}' \cup D_{comp} \times D_{comp}' \cup D_{out} \times D_{out}'$ ,  $R_{jk} \subset D_j \times D'_k$  be relations,  $j, k = 1, \dots, n$ . If

$$RF'_k \subset \bigcup_{j=1}^n F_j R_{jk}, \quad k = 1, \dots, n$$

and

$$R_{jk} G'_k \subset G_j R, \quad j, k = 1, \dots, n$$

Then  $R$  is a weak simulation of  $P$  by  $P'$ .

Proof: We have

$$F = \bigcup_{j=1}^n F_j G_j, \quad F' = \bigcup_{k=1}^n F'_k G'_k.$$

Therefore

$$\begin{aligned} RF' &= R \bigcup_{k=1}^n F'_k G'_k = \bigcup_{k=1}^n (RF'_k) G'_k \subset \bigcup_{k=1}^n \bigcup_{j=1}^n F_j R_{jk} G'_k \\ &= \bigcup_{j=1}^n F_j \bigcup_{k=1}^n R_{jk} G'_k \subset \bigcup_{j=1}^n F_j G_j R = FR \end{aligned}$$

and the proof is complete, by (1).

To give substance to these abstract ideas, some simple machines will be described and simulation proofs sketched.

## V. SIMPLE EXAMPLES OF MICROPROGRAM VALIDATION

The S machine is a simple hypothetical computer described by Haralson and Polivka [15] and is similar to the stack machine of Gear [11]. The stack is in main memory, which is an array SMEM of  $2^{24}$  32-bit words. Additional components in the architecture of S include a one-bit switch SW, an index register SX, an instruction counter SCC, and a stack pointer SSTK, each of size 32 bits. The LSS facility vector for the S machine is shown in Figure 1.

There are 27 machine instructions, which are divided into 14 one-address and 13 zero-address instructions. The one-address instructions need an address operand; the zero-address instructions manipulate the stack and require no operand. The basic LSS macro showing the operation of this computer on the architectural level is shown in Figure 2. If the switch SSW is set, the computer is running, so the macro *fetchword* (SCC) gets the next instruction from memory. The instruction word is broken into parts (by parallel operations) for indirect addressing (id), indexed addressing (ix), op code (op) and address (ad). These values are substituted into the instruction fetch part (instrprep and advctr) to determine the correct address and to advance the instruction counter and are also substituted into the instruction execute macro (execinstr). This basic macro calls itself recursively so the computer will run until the switch SSW is turned off.

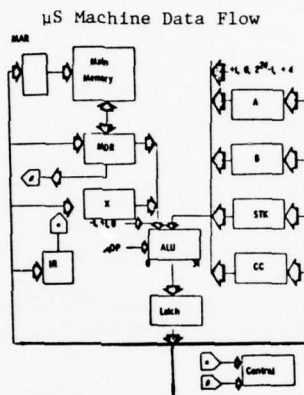


Fig. 7. Data flow for  $\mu S$ .

It has entities MEM, SW, X, CC, and STK corresponding to the components of S. It possesses in addition general purpose registers A and B and a memory data register MDR, each of size 32 bits; a 24-bit memory address register MAR; a five-bit instruction register IR; and a control store CONTROL containing a maximum of 300 sixteen-bit microinstructions.

The original hand proof of simulation had two pairs of domains or stopping points: begin instruction ( $R_0$ ) (isomorphic to an instruction) and machine stopped ( $R_5$ ).

The microcode  $\mu CODE2$  which was proved correct [3] has 172 microinstructions. This code has a major loop beginning when a machine instruction is to be executed. This loop is divided into two sections. In the first, IFETCH, the operands for the current instruction are determined. For zero address instructions the given operand is used; for one address instructions indexing and/or indirect addressing is performed. In the second section, IEXECUTE, the machine instruction operation code is used to calculate a branch to a segment of microcode which performs that instruction. Each of these segments is straight line code, except for the handling of the two shift instructions, each of which contains a simple loop. In spite of the fact that  $\mu CODE2$  was debugged, and simulated, three errors were found and corrected.

A more interesting data flow for the same computer architecture is shown in Figure 8 [20].

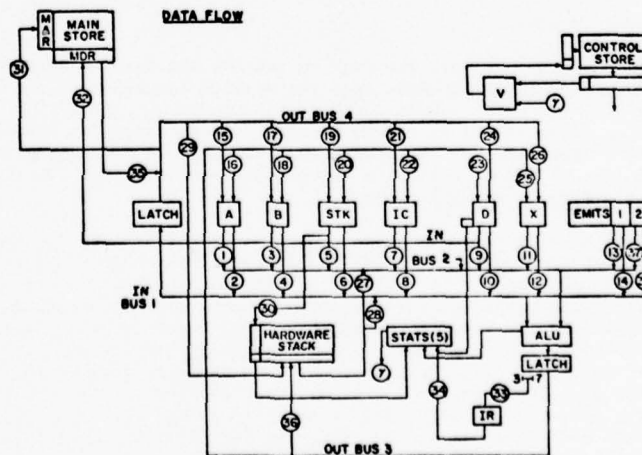


Fig. 8. Data flow for  $\mu S'$ .

The design has the flavor, but not the complexity, of the System/360, Model 40. In addition to the instruction counter (IC) and index register (X) available to a programmer, the microprogrammer can in addition access two general purpose registers (A and B), a special testing register (D), the memory address register (MAR), the memory data register (MDR), a five-bit instruction register (IR), and collections of constants (EMITS 1,2). Note that the stack no longer resides in main memory. The hardware can examine the contents of register D and detect addition/subtraction overflow, stack overflow, and algebraic left-shift overflow; such tests and error checking are performed by the five staticizers (STATS). The read/write operations each extend over three microcycles. The S' machine is horizontally microprogrammed, and the 44-bit microinstruction contains 10 fields as shown in Figure 9. Bits 0 through 11 contain the address of the successor instruction; however, the machine computes the effective address by performing the logical disjunction of the stats and the five low-order bits of naddr. The other fields include a testing field, a memory field, an arithmetic-logical unit function field, two emit fields (which provide needed numerical constants), and one field for each of the four buses.

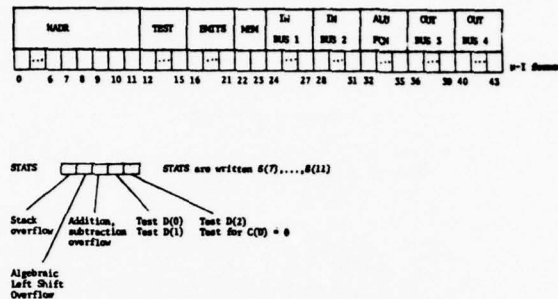


Fig. 9 Stats and microinstruction format for  $\mu S'$ .

Using a self explanatory mnemonic language, we present the first three microinstructions of the microprogram  $\mu P_1$  which emulates the S machine instruction set:

micromemory location	ALU fcn, test	IN Bus 1	IN Bus 2	OUT Bus 3	OUT Bus 4	Memory	NADDR
0	ADD	IC	ONE	IC	MAR	READIN	1
1	ADD	ZERO	FOUR	A			2
2	NOT	ZERO	MASK	B	D	READOUT	5

The first microinstruction fetches the current machine instruction and updates the IC; the instruction becomes available in D two cycles later. The second and third prepare for interpretation of the fetched instruction. The microprogram consists of two parts: IFETCH (micromemory locations 0-15), which fetches the current machine instruction and handles possible indirect addressing; and IEXECUTE (locations 16-90), which executes the instruction. After completion of IEXECUTE, control is always returned to micromemory location 0. There is no temporal overlap between IFETCH and IEXECUTE.

To show that any machine language program in S is correctly implemented by  $\mu S'$ , we must exhibit a strong simulation relation R between S and  $\mu S'$ . We first set

$$\begin{aligned}
 W = \{ \langle u, \langle w \rangle \rangle \in S \times \mu S' \mid & (\text{mem}(u) = \text{mem}(w)) \wedge (\text{sw}(u) = \text{sw}(w)) \\
 & \wedge (\text{stk}(u) = \text{stk}(w)) \wedge (\text{ic}(u) = \text{ic}(w)) \wedge (\text{x}(u) = \text{x}(w)) \\
 & \wedge \text{stack}(u) = \text{stack}(w) \},
 \end{aligned}$$

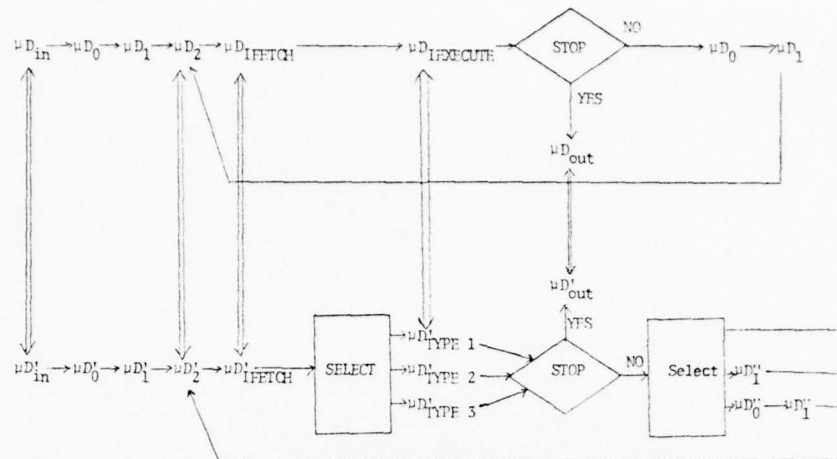
which expresses the conditions we want to hold whenever we examine the machine states at intermediate times. That is, S and  $\mu S'$  should appear the same from the machine language programmer's point of view.

We must now verify the condition  $RF' \subset FR$  for each machine instruction  $i$ . However, by Theorem 4.2, we may break the proof into several pieces. We first show that indirect addressing is performed properly by the relevant microcode in IFETCH for an arbitrary machine instruction [20]. When we enter IEXECUTE, we may encounter straight-line code or possible loops.

The case of straight-line code is easy to handle. A sample proof for the ADD machine instruction, which adds the two top levels of the stack and places the result at the top of the stack, appears in Figure 10. The proofs of the conditions to be verified assume that condition W held at the beginning of the ADD instruction execution. The action of the S machine is symbolically executed in the left hand column. The first instruction gets the word in the architectural stack ( $\text{stack}(u)$ ) which is addressed by the value of the word in the stack pointer ( $\text{stk}(u)$ ) and puts it in temporary (virtual) storage S0. Next 1 is added to the stack pointer, using APL operators, then the next word in the stack is accessed and stored in virtual storage S2. Then the two words are added and the result stored in the stack. The action of the  $\mu S'$  emulation, shown in the right hand column, is guided by the microprogram  $\mu P_1$ . After this emulation, which uses the  $\mu S'$  registers A, B and STK, the verification conditions are that the new values at the top of the stacks are equal. This proof is straight forward for a human.

<u>S-begin</u>	<u>μS'-begin</u>
s0: stack(u) [21stk(u);]	A : stack(w) [21stk(w);]
stk(u) ← s1: (32ρ2)τ(2*32)   1	stk(w) ← n1: (32ρ2)τ(2*32)   (21
+ 21stk(u)	stk(w)) + 21(31ρ0), 1
s2: stack(u) [21stk(u);]	B : stack(w) [21stk(w);]
s3: (32ρ2)τ(2*32)	stack(w) [21stk(w);]
(21s0)+21s2	← n3: (32ρ2)τ(2*32)
stack(u) [21stk(u);]	(21n0)+21n2
+ s4: s3	
Verify: (1) s1 = n1	
(2) s4 = n3	
<u>Case I: overflow</u>	
sw(u) ← s5: (2ρ2)τ2	sw(w) ← n4: (2ρ2)τ2
<u>S-end</u>	<u>μS'-end</u>
Verify: (3) s5 = n4	
<u>Case II: no overflow</u>	
<u>S-end</u>	<u>μS'-end</u>
Verify: empty	



Figure 11. Equivalences between two micromachines  $\mu M$  and  $\mu M'$ .

In carrying out these formal proofs by hand, it became apparent that the individual parts of such proofs were not of great complexity, but that the main impediment to human proofs was the generation and organization of these many separate parts. Since this number of parts increases with the size of the implementation being verified, it became clear that some automated aid would be necessary. In the following we shall describe the Microprogram Certification System, (MCS), an interactive system, designed to aid in proving simulation between programs.

#### VI. THE MCS SYSTEM

MCS (for "Microprogram Certification System") is written in LISP and provides interactive aid for proving that a stated relation between two machine descriptions is a simulation. A critical problem in the hand proofs of microprogram correctness [3, 19, 20] was to assure that all pairs of simultaneous paths (from one component of the simulation relation to the other) were taken and all theorems generated and proved. In MCS this bookkeeping function is embodied in an interactive, goal-directed, and problems independent supervisor [4]. From the user's point of view, this supervisor provides a uniform interface through which he manipulates a tree of goals. Using a set of standard commands, he controls the direction of the proof of simulation and observes its progress. The components of MCS which do the actual work of proving simulation - the path tracer, simplifier, verification condition generator, theorem prover, etc. - are invoked by the user through this supervisor.

The data structure upon which the MCS supervisor operates is an AND goal tree, and the user manipulates this tree in a problem-reduction fashion [4,27]. Initially the tree is empty; the user supplies a first goal, which embodies the problem which he wishes to solve, and which becomes the current goal, or focus of attention. Each goal consists of a flag indicating whether or not the goal has been achieved, a goal class or list of functions for attacking the goal, and a pattern describing what the problem represented by the goal is. To act upon the current goal, the user invokes one of the rules in the goal class. This LISP function is then called with the elements of the pattern as arguments. The rule may achieve the goal directly or may generate one or more subgoals of the goal, the achievement of all of which is sufficient to establish the original goal. If subgoals are generated, the first of them becomes the current goal. If a goal is achieved, its nearest unproved brother becomes current; when all sons of a goal are achieved, that goal is marked as achieved and the search for an unproved goal is repeated. The original problem is solved when the initial goal is achieved.

In addition to the two operations of adding a new goal by specifying its goal class and pattern, and invoking a function upon the current goal, a new current goal from the tree may be selected by specifying its index number. (Goals are indexed in a Dewey-decimal fashion.) This flexibility allows partial solutions to problems.

The initial goal entered by the user in proving simulation between programs indicates that the problem is to prove that a given simulation relation holds between two given machine descriptions. The pattern of such a goal is a list of five items: The state vector and macro library of each machine, and the simulation relation (in the form specified above). These items are normally read from a file. The class of such an initial goal, called SIMULATE, contains a single rule, RSPLIT, which generates one subgoal for each of the components of the simulation relation. Each of these subgoals, of class TRACEIT, has in its pattern the state vector, control tree, and macro library for each machine, a current predicate list of assumptions, and the complete simulation relation. The control tree of each machine is taken from the component of the simulation relation to which the subgoal corresponds. The state vector is formed by prefixing "S" to the name of the machine component. The predicate list is created from the predicates given in the stopping conditions and simulation conditions of the simulation component, although some of these predicates may not appear explicitly in the predicate list but may be reflected in the initial values of the state vector quantities. Figure 12 shows the element of the pattern of a goal of class TRACEIT, generated by RSPLIT from the simulation component of Fig. 3.

To achieve each of the goals generated by RSPLIT, the user must show that starting from each state, and running both abstract machines to their next stopping points, the simulation conditions of the pair of points reached hold. The proof is completed by showing that the theorems of [21] are satisfied; then, since the initial conditions are true, the output conditions are valid. Since there is a very large number of possible execution sequences of the two machines (corresponding to the number of possible actual values of the state vector quantities), symbolic execution [4, 7, 16] is used. Invocation of the rules TRACEM or TRACEUM (for the algorithmic level and the implementation level, respectively) initiates the symbolic interpreter on the control tree and state vector of the goal.

(SMEM . \$MEM	}	state vector, control tree, and macro library of specification level		
SSTK . \$SSTK				
SX . \$SX				
SCC . \$SCC				
SSW . \$SSW				
<u>exec-pgm</u>				
SLIB	}	state vector, control tree, and macro library of processor level		
(MSSTK . \$SSTK				
MSCC . \$SCC				
MSA . \$MSA				
MSB . \$MSB				
MSX . \$SX				
MDR . \$MDR				
MAR . \$MAR				
MSSW . \$SSW				
MSMEM . \$MEM				
CS . microcode				
CSAR . 000000000001				
IR . \$IR				
<u>exec-mpgm</u>				
MSLIB				
NIL		predicate list		
R		simulation relation		

Figure 12. A Goal of Class TRACEIT

The abstract, or symbolic, interpreter carries out the symbolic execution of each machine by employing the algorithm in Section II. A more detailed description of the path tracing portion of MCS is given in [22]. As discussed in Section II, at each step of the algorithm either the control tree s-control is modified, or either local or global variables are appropriately modified. At each step of the simulation relation is checked to determine whether a stopping point has been reached; when it is, interpretation halts.

This procedure is complicated by the fact that the state vector elements have symbolic, rather than actual, values. This chiefly affects two aspects of the interpretation. First, the expressions in assignment statements are symbolic, and cannot usually be evaluated to numeric or boolean values. Second, when predicates are encountered in expanding macros, they cannot always be evaluated to "true" or "false".

Both of these problems are partially solved by a simplifier, which performs the symbolic computation done in MCS. Whenever the interpreter encounters an assignment statement, a predicate in a conditional, or the passing of arguments to a macro, the simplifier is invoked to "evaluate" APL and logical expressions by returning to a simpler form if possible. In addition, the simplifier is called by the theorem prover in attempting to prove generated verification conditions and to determine at each step in the symbolic execution whether a stopping point has been reached. This ubiquity of points at which simplification is required makes this the component of MCS in which the most time is spent.

The simplifier in symbolic execution must provide an extended semantics for each APL and logical operator encountered. This in general agrees with the normal semantics when the operands are actual values; e.g., the result of evaluating of  $3 + 2$  remains 5. The extension is needed because operands may now be symbolic and of various forms; each such operator and form of operand may require a different semantic rule describing the action to be taken. For example, the result of  $\text{exp} + 0$  is  $\text{exp}$ , and the result of  $(-\text{exp}) + \text{exp}$  is 0, whatever the form of  $\text{exp}$ . Two desirable attributes of such an extended semantics are to perform as much of the actual computation as possible, and to eliminate irrelevant parts of expressions.

In the MCS system the simplifier consists of a set of rules for each operator, invoked by pattern matching in a manner similar to the QA4 system [29]. They are domain-oriented, applying to a subset of APL and logical expressions, and are designed to be easily modified by the user. Each consists of a pattern, containing variables to be matched against parts of an expression to be simplified and perhaps constrained by specified predicates; and a body, which, when instantiated by the variable values resulting from the pattern match, is the simplified value of the expression

Examples:

$$(A,B)[M+\uparrow N] \rightarrow B[M-(\rho A)+\uparrow N], \text{ if } \begin{array}{l} (\rho A) \leq M \text{ and} \\ (\rho B) \geq (M-(\rho A)+N) \end{array}$$

is a rule with variables A, B, M, and N, pattern (A,B) [M+\uparrow N], body B[M-(\rho A)+\uparrow N], and two constraints on the variables. (These constraints must be passed to the theorem prover.)

A+0-A has pattern A+0, body A, and no constraints on the single variable A.

When an expression is passed to the simplifier, each argument of the outermost operator is first simplified. Then, the list of rules for that operator is searched. If a rule with pattern matching the expression is found, the expression is replaced by the (instantiated) body of the rule, and the entire process is repeated. When no applicable rules are found, the expression has been simplified. (This process is similar to that followed in the SCRATCHPAD system of Griesmer, Jenks, and Yun [13].) Thus further extensions to the semantics of an operator may often be made by the addition of rules for that operator.

In MCS, newly created expressions are simplified at once to prevent the propagation of unsimplified forms. At present MCS has over 400 rules for the simplification of APL and logical expressions. Because we are using APL operators to describe manipulations of registers, memory, and other machine components, a large number of the simplification rules in our system relate to the APL indexing and concatenation operators over vectors and matrices. Assignments made to particular locations in memory are effected by forming a new APL expression for the memory by concatenation. To assure the correctness of the simplification rules for these arrays (which are not always obvious), many of the identities which they express have been proved to hold, using More's axiomatization of array theory [26] based on an APL-like language.

When the interpreter expands a macro with predicates, it must attempt to prove each predicate true or false (using the current state vector values and predicate list) before it can proceed. But this is in general impossible, since the predicates contain symbolic values. If a predicate can be shown true, the interpreter continues with the expansion. Otherwise, since all possible paths must be followed, the interpreter stops at this point and generates one subgoal of type TRACEIT for each predicate that cannot be shown contradictory. In such a subgoal, the predicate (after instantiation by the state vector) is added to the predicate list. Often, however, predicates can be reflected in the state vector values and removed from the predicate list entirely. For example, if the branch corresponding to predicate \$X=0 is to be taken, symbolic value \$X can be replaced everywhere (including possible occurrences in the state vector of the machine not being executed) by zero.

Assertions about portions of registers such as \$Y[12]=1, can also be reflected in this way, here by replacing \$Y by 1,1,\$Y[2+30], if \$Y is a 32-bit vector. Care must be taken in making such substitutions to avoid loss of information. In case of an equation of the form \$X=f(\$X), substitution for \$X can be made if and only if f(\$X)=f(f(\$X)) can be shown [22]. When several simultaneous assertions are made, such as when assuming the stopping and simulation conditions before path tracing, the process is slightly more difficult. Predicates other than equalities must be retained on the predicate list. Reflecting of predicates in the state vector helps path tracing by making future predicates easier, or even trivial, to prove, and by considerably simplifying the expression for the state vector quantities.

Normally, the abstract interpreter is applied first to the specification description (TRACEM) until a stopping point is reached, and then to the processor level (TRACEMUM). In each case, the interpreter constantly checks to see if one of the stopping points defined by the components  $R_1, \dots, R_n$  of T has been reached; if so, a subgoal of type TRACEIT is generated. Note that all loops or potentially infinite expansions in the machine description must have associated stopping points in some component  $R_i$ . In the event that such a stopping point were omitted, the interpreter would repeatedly generate new subgoals at a decision block in the loop, or it would perform macro expansions and contractions endlessly.

When a goal is generated for which both descriptions have reached stopping points, the verification condition generator is invoked through the rule GENVC, still a member of class TRACEIT. This verification condition generator has two main functions. The first is to verify that if  $R_i$  and  $R_j$  are the simulation components reached by tracing the paths of the two abstract machines to stopping points, then  $i = j$ ; i.e., another point of correspondence in the simulation relation has been reached. The second is to instantiate values from the two state vectors into the list of simulation conditions (simultaneous verification conditions) given in the specified component  $R_i$ . GENVC generates a goal of class THMPRV for each of these instantiated conditions, having as its pattern this theorem and the predicate list of the current goal. Often, because of the continual simplification and incorporation of equalities into the state vector during interpretation, these theorems are equalities between identical expressions. An example of a generated theorem appears in Figure 13.

$$((8\phi 0), ((32\phi 2) \uparrow (2\uparrow \$SX) + 2\uparrow \$SMEM[2\uparrow \$SCC[8+\uparrow 24]; 8+\uparrow 24]) [8+\uparrow 24]) [\uparrow 8] = 8\phi 0$$

Figure 13.

Sample theorem generated by GENVC and proved by the theorem prover.

Generally, the rule PROVEIT is used on goals in THMPRV. This invokes the theorem prover and simplifier on the theorem of the pattern (see below). The theorem prover is embodied in simplification rules for logical expressions, involving the relational operators = and  $\leq$  and the logical connectives  $\wedge$ ,  $\vee$ , and  $\sim$ . This theorem proving may be viewed as an extension of simplification. If a theorem simplifies to "true", then the goal is achieved (and theorems are the only goals generated in proving simulation that can be achieved directly, without generation of subgoals.) If the theorem cannot be proved by the theorem prover, its simplified form is the theorem of the pattern in a single generated subgoal. Two other rules in class THMPRV may then be invoked. ANDSPLIT is applicable if the theorem to be proved is conjunction - it



generates a separate subgoal for each conjunct, also in class THMPRV. CASESPLIT can be used to split the given theorem into two cases on the basis of a predicate supplied when invoking the rule. The given predicate is appended to the predicate list of one of these subgoals and its negation to the other. The user may then proceed to split into further cases or to attempt the new subgoal directly using PROVEIT. Then, since each subgoal contains more information, the theorem prover may be able to establish a theorem which it was not powerful enough to establish before.

Though the necessity for theorem proving in MCS is most visibly apparent in the need to prove the verification, it is also needed at various other points, such as in deciding predicates at branch points and in determining whether a stopping point has been reached.

In the case of equalities between APL expressions, MCS relies heavily on the fact that expressions which can be proved equal have the same simplified form. The present set of rules certainly is not complete in the sense that such a canonical form is assured in all cases. However, the incorporation of equalities into the state vector does assure that both sides of an equality are expressions over the same set of symbolic constants and often eliminates the need for references to the predicate list of assertions to resolve equalities.

If all the theorems on all branches are proved true, the supervisor will mark the top level goal as achieved, and simulation will have been shown. The goal tree itself is a record of the proof that whenever the stopping points are reached, the simulation conditions are satisfied. Errors in the microcode (or in the formal descriptions) are detected by being unable to prove theorems at the leaves of the goal tree; by tracing back toward the root, information about the particular instruction or place in the description at which the error occurred can be obtained.

Figure 14 shows a portion of the goal tree (corresponding to operandfetch) for the automated S-machine experiment, described in the next section.

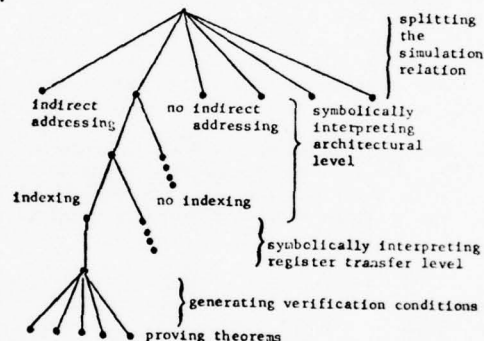


Figure 14.

## VII. THE AUTOMATED S-MACHINE EXPERIMENT

Portion of Goal Tree for S-Machine Experiment

This simple machine was described in Section V, and the formal hand proof of the validity of the microcode  $\mu\text{CODE 2}$  sketched. The formal descriptions of both machines  $S$  and  $\mu S$  the microcode  $\mu\text{CODE 2}$ , and the simulation relation  $R$  were entered in the S/370 model 168 and analyzed by a preprocessor. This program forms state vectors  $\text{SSV}$  and  $\mu\text{SSV}$ , which contain the elements of the respective facility vectors of  $S$  and  $\mu S$  along with the initial symbolic values. The permanent value of  $\text{CS}$  is  $\mu\text{CODE 2}$ , and the initial value of each remaining component is its name prefixed by a dollar sign.

As pointed out above, natural places to choose for stopping points depend upon the architectural description (in this case the  $S$  machine), and upon the structure of the microcode. In the original  $S$  description [3], each machine instruction was described separately; in the microprogram, portions of the code were shared by several instructions. To make the two descriptions more similar new macros were added to the  $S$  description to consolidate the work previously done in each instruction. These changes produced a second architectural description which was easily shown by hand to simulate the initial one. The proof was completed by showing a simulation relation between the new description and the register-transfer description  $\mu S$  and using the transitivity of simulation.

The original hand proof of simulation had two pairs of domains or stopping points: begin instruction ( $R_0$ ) (isomorphic to end instruction) and machine stopped ( $R_6$ ). The change in description allowed the addition of the new domains  $R_1$  (zero address instruction calculation),  $R_2$  (one address calculation),  $R_3$  (left shift simultaneous verification condition), and  $R_4$  (right shift simultaneous verification condition). The number of paths to be traced was reduced by this addition, and shorter paths and easier theorems resulted.

All of the pairs of paths emanating from the six components of the simulation relation were traced, and the following results obtained.

Corresponding pairs of simulation paths traced:	45
Verification conditions generated:	236
Verification theorems proved	
Without interaction:	222
With interaction (splitting into two cases):	2
Verification conditions not proved because $\mu P$ Errors:	12



The system discovered the three errors that had been found previously by hand. However, an error in address generation was found. Direct addressing, indirect addressing and indexed indirect addressing stored an address as eight zeros followed by 24 address bits, while indexed addressing stored an address as eight random bits followed by the 24 address bits. This is a new error not discovered in the hand proof. The 24 address bits are correct, but the bits from 0-7 in the word are not masked to zero and depend upon the contents of the index register.

The corresponding corrections were made in the  $\mu P$ . Then MCS traced 45 paths, generated 236 verification conditions and proved 236 theorems, with human aid to prove two. Thus the new  $\mu P$  was validated.

#### VIII. THE HTC EXPERIMENT

MCS is now being applied to a computer architecture which is more complicated in many ways than the S-machine. The HTC (for Hybrid Technology Computer) is a product of the IBM Federal Systems Division at Huntsville for space flight applications [32]. Its architectural specifications require it to support the System/360 standard instructions (no decimal or floating point operations) with sixteen 32-bit general purpose registers, 16 to 64 K bytes of main storage, and a single I/O channel for three types of I/O. This architecture is implemented in a machine having a 1K memory of 64-bit microinstructions, a 64x16 bit read-only memory for instruction decoding, an ALU and two input multiplexers, three working registers, and a 16-bit wide data path.

Part of the increased difficulty with the HTC lies in formalizing the description of the architecture and register transfer levels. Architecturally, the HTC differs from the System/360 in several ways. In addition to emulating the specified instruction set, the HTC microcode also implements interrupt and I/O handling. Thus the formal architectural description must be pieced together from the System/360 documentation and other HTC information. Also, in the machine instruction definitions, parts of the specifications (such as condition code settings) were left intentionally ambiguous to allow implementation on various S/360 models; the architectural description must allow for this vagueness. Our present approach has been to use the APL ? operator in these cases, and to include specific simplification rules for handling expressions with this operator. For example, if the two bit condition code is specified to be unpredictable in an instruction, it will have a value of ?2p2 at the end of that path. The microcode will give the condition code some specific value, and the theorem prover must be able to answer "true" for the equality of the two.

The single 16-bit wide I/O channel of HTC has been modeled by an array of 16 columns and an infinite number of rows (see [26]). Input from the channel is done by taking the first row of the array as the input word; output is done by concatenating rows onto the end of the array. Thus the words sent to the channel in each machine can be compared when a stopping point is reached; their equality is necessary for the proof of simulation.

In the HTC, the register-transfer level has a 16-bit data path with 16-bit register and storage locations; the architectural level is 8-bit byte-oriented. For example, each 32-bit general purpose System/360 register is stored in two non-contiguous locations in the HTC scratch pad memory, and the 16Kx8 bit main store of the architectural description is implemented in an 8Kx16 bit memory in the hardware. Thus the APL expressions stating the equalities between the machine components are more complex than in the S-machine, where the architectural components are a subset of those at the register transfer level.

The HTC register transfer level description describes how the microcommands in a microinstruction act to modify the values in the registers and storage in a real computer. Since the validity of the simulation proof depends upon the accuracy of this description, only hardware entities are made part of the micro machine state vector. However, macros should be written to be direct, and perform calculations in a straightforward manner so that simulation speed is improved. These requirements conflict frequently. In the HTC, the ALU is used during each microcycle, with inputs selected according to a complicated combination of micro commands, and the output is similarly distributed. For clarity these inputs and the outputs were represented as quantities in the state vector rather than in one large macro. Even so, processing was slow. Examination of the machine microcode showed that frequently the ALU simply assembled and passed data without performing arithmetic or logical operations. By testing the few fields indicating such constructions and performing them directly the speed of operation of the micro machine was considerably improved. While this type of analysis can be done by programs, it indicates that for efficient microcode validation a good knowledge of both the microcode and the data flow is presently required.

The HTC is a real machine, so the methods by which it physically starts (stops) and the consequences of its timing conditions and asynchronous actions must be modeled. Starting and stopping concerns both the architectural and register-transfer levels while the timing conditions primarily affect the register transfers. Since the HTC is an airborne computer in the Space Ultra-reliable Modular Computer (SUMC) family, its connections with the outside world are controlled by microprogrammed Test Support Equipment (TSE) which simulates an HTC channel and also has lines directly connected to the HTC. The HTC channel and accompanying interrupts were mentioned earlier. The control lines (activated by buttons on the TSE console) and their functions are shown in the following table:

Architecture	Register Transfer	Function
Power On/Off	Clock Start/Stop	Activate the basic clock cycle.
Reset	Hardware Reset	Set some registers and flip flops to known states. Then, by microprogram, set the remaining basic registers so the machine architectural and support functions may be initiated.

Soft-Stop

STOP

Stop performing architectural functions  
and return to the reset condition ready to  
receive external commands

The basic architectural functions which may be initiated are IPL and Read Paper Tape (for program entry). The support functions are the usual maintenance functions - register display, stop and display registers when a particular address (in main storage or in control store) is reached, clear and test main store, and display main storage or control storage locations. All of these functions are initiated by the TSE and performed under microprogram control using the HTC channel. Since these functions are implemented in microcode, an architectural level description for them must be given and a simulation between the two levels specified and proved.

The basic HTC machine cycle is 550 ns and the main storage cycle is 700 ns. One microinstruction is performed per machine cycle. The action of the registers is asynchronous, but the microinstruction actions are performed roughly sequentially except for interactions between the main store and the computer. The action of the registers is determined by sequential leaves on the HTC control tree, as was shown in the S-machine example. The timing interaction between the computer and main storage is considered only to ensure that the contents of affected registers, the Storage Data Register and the Instruction Register, are valid. All timing is thus relative to the microinstruction execution, so a pseudo counter, CTR was introduced. Whenever a microinstruction is read, CTR is incremented. Each of the two registers affected by timing is replaced by two pseudo registers. The first contains the usual register contents and this register is set as if there are no timing restrictions. The second pseudo register contains the contents of CTR at the time the first is set. When one of these registers is accessed, the value of the second register is compared with the contents of the CTR, and if the difference is too small a timing error is signaled. The Instruction Read Only Memory (IROM) initiates the only asynchronous action which may cause difficulties. When a word from main storage is put in the left half of the instruction register and the register is valid, the eight bits corresponding to the op code go immediately to the IROM as address. This register too is represented by two pseudo registers but the second register is updated by CTR+1 instead of CTR.

Finally the size of HTC as compared with the S-machine makes necessary a more complex simulation relation, more paths to be interpreted, and more difficult theorems to be proved. The HTC implements an instruction set three times larger than that of the S-machine, and has eight times as many microinstructions, each of which is four times larger.

At present the HTC architectural and register transfer levels have been formally described, the simulation relation has been partially formulated, portions of the microcode have been proved correct, and some errors have been detected. Several of these have been subtle errors which are difficult to detect using test cases. For example, one of them would occur only when fetching a 16-bit instruction from the last halfword of addressable memory. Another more serious flaw was found in the implementation of the BALR or branch-and-link instruction: in cases where the link information was to be stored in the same register containing the branch address, the branch address was erroneously lost. All of these errors were not found by testing or simulation: They were detected because of unprovable theorems being generated in the certification process. The goal structure of MCS enabled the user to go directly to a small segment of microcode and correct the error.

#### IX. CONCLUSIONS

Our aim in the SVAE system has been not only to obtain proofs of correctness, but also to detect and correct errors. The structuring of the proof into goals imposed by the supervisor aids in this detection. Indications of error, such as being unable to prove a theorem, failing to reach an expected stopping domain and unexpected branches, occur when the user invokes a rule in a particular subgoal. Examination of the branches of the goal tree leading to the error indication provides some information as to the cause of the error.

The problem reduction approach provided by the MCS supervisor has several advantages over both hand-proof techniques and more straightforward automated methods. Most important of these is that no path to be traced can be over-looked; the supervisor always returns to the unproved goals. Also, the goal tree provides a record of the way in which the proof was developed. Finally, the capability to decide the order in which the goals are to be manipulated permits partial proofs; certain sections of program or microcode can be verified even before correctness criteria, in the form of simulation relations, have been developed for other parts.

The automated parts of our system were found most useful for coordinating parts of the proof, simplifying symbolic APL expressions, and running the symbolic interpreter. The immediate simplification of expressions and processing of predicates encountered contributed greatly to the simplicity of the final theorems generated; this was especially helpful in view of the interactive nature of the system. Also found valuable were the ability of the user to control the proof direction through the supervisor and the rules for processing them while the proof is in progress.

Of course, the human contribution to the SVAE automated validation of implementations consists of more than interaction with the MCS system. Describing the abstract machines which embody the system specifications is by no means trivial, especially when the English "principles of operation" are vague. Also specification of the simulation relation requires some understanding of how the program being verified works (location of loops, etc.), though use of MCS to interpret symbolically a single program may provide aid in developing simulation conditions and stopping points. The judicious choice of stopping points can greatly reduce the number of the paths which must be followed and the theorems which must be proved.

The successful detection and correction of errors in a small microprogram using SVAE and our progress toward the verification of an actual microcoded implementation, have confirmed our beliefs that computer aid in validating the design of computer systems is needed and valuable, and that the notion of simulation between programs facilitates this automation.

## BIBLIOGRAPHY

- [1] Ashcroft, E. A. and Z. Manna, Formalization of Properties of Parallel Programs, *Machine Intelligence* 6 (1970) pp. 17-41.
- [2] Bell, C. G. and A. Newell, "Computer Structures: Readings and Examples," McGraw Hill, 1971, P. 562.
- [3] Birman, A. On proving correctness of microprograms, *IBM J.R.&D.* 18, 4 (May 1974), 250-266.
- [4] Birman, A. and W. H. Joyner. A program reduction approach to proving simulation between programs, RC5525, IBM T. J. Watson Research Center, (July 1975). To appear IEEE Transactions on Software Engineering.
- [5] Buckingham, B. R. S., W. C. Carter, W. R. Crawford, G. A. Nowell, The Controls Automation System, Sixth Symposium on Switching Circuit Theory and Logical Design, 1965.
- [6] R. M. Burstall, An Algebraic Description of Programs with Assertions, Verification, and Simulation, Proceedings of an ACM Conference on Proving Assertions about Programs, Las Cruces, New Mexico, 1972, p. 12.
- [7] Deutsch, P., An interactive program verifier, Ph.D. Thesis, U. of Cal., Berkeley, 1973.
- [8] Ellozy, H. A. An approach to the synthesis of equivalence relations for program verification, to appear.
- [9] Falkoff, A.D., K. E. Iverson, E. H. Sussenguth, A Formal Description of S/360, *IBM Systems Journal*, Vol. 3, No. 2, 1964.
- [10] Floyd, R. W. Assigning meaning to programs, *Proc. Symposia in Applied Mathematics* 19, 1967, 19-32.
- [11] Gear, C. W. *Computer Organization and Programming*, McGraw Hill, 1969, p. 43.
- [12] Good, D. I., R. L. London and W. W. Bledsoe. An interactive program verification system. *IEEE Transactions on Software Engineering*, SE-1, 1 (March 1975), 59-67.
- [13] Griesmer, J., Jenks, R., and Yun, D., *SCRATCHPAD User's Manual*, RA 70, IBM T. J. Watson Research Center, (June 1975).
- [14] Goguen, J.A., Jr., "On Homomorphisms, simulations, correctness and subroutines for Program and Program Schemes," *IEEE 1972 Switching & Automata Theory Symposium*, p. 58.
- [15] Haralson, K., and R. Polivka. Microprogram training - an APL application. Proc. 4th Int. APL User's Conf., 1972.
- [16] King, J. C. A new approach to program testing. *Proc. Inter. Conf. on Reliable Software*, Los Angeles, (April 1975), 473-481.
- [17] Landon, P.J. The Mechanical Evaluation of Expressions, *The Computer J.*, 6, (1964) No. 4 pp. 308-320.
- [18] Lee, J.A.N., Computer Semantics, Van Nostrand, 1972, p. 3.
- [19] Leeman, G. B., W. C. Carter and A. Birman. Some techniques for microprogram validation. *Proc. IFIP*, Stockholm, (August 1974), 76-80.
- [20] Leeman, G. B. Some problems in certifying microprograms, *IEEE Transactions on Computers* C-24, 5 (May 1975), 545-553.
- [21] Leeman, G. B. A Method for Proving Equivalence of Programs, RC5022, IBM T. J. Watson Research Center (September 1974).
- [22] Leeman, G. B. Symbolic Path Tracing in MCS, RC5642, IBM T. J. Watson Research Center, (September 1975).
- [23] Lehman, M.M. Microprogramming trend considered dangerous. *C.ACM* 18, 6 (June 1975), 358-360.
- [24] McCarthy, J., "Towards a Mathematical Science of Computation", *In Information Processing 1962*, North Holland, 1963, pp. 21-28.
- [25] Milner, R. An algebraic definition of simulation between programs. Proc. Second Inter. Conf. on Artificial Intelligence, London
- [26] More, T. Axioms and theorems for a theory of arrays, *IBM J. R&D* 17 (1973), 135-175.
- [27] Nilsson, M. J. Problem Solving Methods in Artificial Intelligence, New York; McGraw-Hill, 1971.
- [28] Rosen, B. K. Tree-manipulating systems and Church-Rosser theorems, *J. ACM* 20, 1 (January 1973), 160-187.
- [29] Rulifson, J. F., J. A. Derkson and R. J. Waldinger, QA4: a procedural calculus for intuitive reasoning. TN-73, Stanford Research Institute (November 1972).
- [30] Sethi, R. Testing for the Church-Rosser property. *J. ACM* 21, 4 (October 1974), 671-679.

- [31] Suzuki, N. Verifying programs by algebraic and logical reduction. Proc. Inter. Conf. on Reliable Software, Los Angeles (April 1975), 473-481.
- [32] Technical Description of SUMC Computer Model 2B, IBM Doc. #7CW 00013, Dec. 31, 1975.



## SYSTEM INTEGRITY BY USE OF SELF-DIAGNOSING FAILURE DETECTION

R. Onken

Deutsche Forschungs- und Versuchsanstalt  
für Luft- und Raumfahrt e.V. (DFVLR)  
Institut für Flugführung  
3300 Braunschweig, Flughafen  
West-Germany

Summary

A definition of the system integrity function is given in order to enable comparative evaluation of different design approaches for digital flight control systems with respect to the system integrity and mission survival. Two main failure detection methods are analytically investigated with respect to the integrity function, the passive failure detection and the selfdiagnosing active failure detection. The HFB 320 experimental system is briefly described as an example of the selfdiagnosing active failure detection method.

1. Introduction

During the last decade a number of fly-by-wire experimental systems, both analog and digital, have been developed and flight-tested in several countries (e.g. ref. 1, 2, 3), most with the intention of demonstrating the feasibility of flight control by electronic means without mechanical back-up. Of course, other aspects such as new control law concepts were also explored, but the main concern has been to ensure the required levels of *integrity* for the hardware and software of the control system. Most of these systems differ considerably in the design philosophy used to ensure the required levels of integrity. But so far not enough theoretical work has been done to allow sufficiently precise quantitative comparison of system integrity levels between different design philosophies. In particular, this is true in the case of digital systems.

This article attempts to contribute to the development of easily adaptable methods for analytically comparing designs. Although more general investigations were completed only those for the electronic part of the system hardware and its software are described here. In that context, the criterion for the mission abort, which is derivable from the loss of integrity is a reasonable quantity to look at. This criterion might be the maximum number of detected failures which are permitted, but it might also be in more general terms the maximum permitted probability of total loss within the mission or operation time of the system. Here the latter is used by applying the integrity function. This function represents the actual status with respect to the integrity. It contains the basic information from which the extrapolation of the probability of total system loss during the remaining mission time can be derived.

There are two reasonably cost effective means to ensure system integrity for flight control systems. First redundancy, and second the addition of monitoring and switching mechanisms in order to make use of the available redundancy in case of a malfunction. The potential of the redundancy concept is well established, whereas the possible potentials of the malfunction detection mechanism have not been explored thoroughly, particularly regarding its effect on the integrity or, for the mission abort criterion, on the probability of total system loss. Therefore some quantitative relationships will be outlined in the following showing the effect of the quality of the malfunction detection mechanism on the integrity function. Additionally, a brief description will be given of the HFB 320 experimental digital fly-by-wire system. For this system an unusual approach to malfunction detection called selfdiagnosing active failure detection (SAFD) has been mechanized.

2. Principles of malfunction detection

The *fundamental detection principle* which must be applied in all cases is the comparison of output signals from functionally equivalent processing units. These output signals are independently derived from the same input signal and are dependent on the status of the process. Discrepancies at the comparator indicate a malfunction. Next the *test principle* is applied. The way of applying the test principle determines how long it takes for a malfunction to become evident. The objective of the test method is to ensure that the input signal adequately exercises all components in the signal processing unit. The higher the test frequency for all processing states, the faster the detection of any malfunction at the output.

On this article two different test schemes are evaluated. One is independent of the process and its status and the other is dependent on the status of the process. In the latter case the test signal is simply the normal, unmodified input signal. This is controlled by the statistics of the process and not by the detection mechanism. This case will be defined as *passive failure detection*, as opposed to *active failure detection*. For the latter the test is carried out independently by periodical checks of all states of each system component. In most cases the active failure detection occurs simultaneously with the process i.e. no process interrupts are necessary for testing (ref. 3, 4). Although real versions of failure detection are not necessarily mechanised exactly in one of these two ways, we will use these in the following for the purpose of covering as wide a spectrum as possible of similar systems.

### 3. Failure detection performance - integrity function

It is well known from the literature (ref. 4, 5, 6) that the stochastic process of failure occurrences in electronic components is described as a stationary Markov process, where the density function  $f(t)$  is exponential:

$$(3.1) \quad f(t) = \lambda \cdot e^{-\lambda t}$$

and the coefficient  $\lambda$ , known as failure rate, is considered as a constant. The distribution function  $F(t)$ , which is the probability of a failure event  $t_F$  prior to  $t$ , defined as

$$F(t) = P \{t_F \leq t\},$$

represents the integral of the density function:

$$(3.2) \quad F(t) = \int_0^t f(\tau) d\tau = 1 - e^{-\lambda t} = 1 - R(t),$$

where  $R(t)$ , known as the reliability function, is the probability that no failure occurred up to  $t$ .

An important property of this type of stochastic process is that the distribution function for a certain time interval  $(t_a, t_a + \tau)$  does not depend on the time prior to  $t_a$ ; it depends only on the width  $\tau$  of the time interval. This can be shown by the following relationship (ref. 5, 6, 7): If  $R(t_a, t_a + \tau)$  is the conditional probability that no failure will occur within the interval  $(t_a, t_a + \tau)$  assuming also that no failure has occurred before  $t_a$ , then (see fig. 3.1)

$$(3.3) \quad R(t_a, t_a + \tau) = \frac{R(t_a + \tau)}{R(t_a)} = \frac{e^{-\lambda(t_a + \tau)}}{e^{-\lambda t_a}} = e^{-\lambda \tau} \\ = 1 - F(t_a, t_a + \tau).$$

For an electronic component with failure detection capability, from which at the actual time  $t_a$  correct information has been obtained that no failure has occurred so far, then again  $R(t_a, t_a + \tau) = R(\tau)$ . As can be concluded from fig. 3.1, the value of  $R(t_a, t_a + \tau)$  can be considerably above that of the original  $R(t)$ . In that case,  $R(t_a, t_a + \tau)$  can even be 1 for  $\tau = 0$ , i.e. full integrity has been established. Of course, it has been assumed that the failure detector worked ideally and corresponding to its message no failure has in fact occurred. That is not true for most of the applications, and  $R(t_a)$ , which is  $R(t_a, t_a + \tau)$  with  $\tau = 0$ , is less than 1. Therefore shortcomings in the performance of the failure detector directly increase the uncertainty of the integrity measurement time  $t_a$ . As a measure for this uncertainty,  $R(t_a)$  is used in the following, which is called the integrity function. The integrity function contains what is known up to  $t_a$  and it can be understood as a kind of transition function for the probability of surviving the rest of the mission.

In order to evaluate the effect on the integrity function on the different methods of failure detection, the degree of redundancy, which of course also effects the overall integrity is first eliminated. This can be achieved by the assumption that sufficient spare units are available to repair each failure which has been detected.

With this assumption, the integrity function  $R(t_a)$  can be derived by considering a single channel unit only and by assuming that no failure has been detected up to  $t_a$ .

### 4. The integrity function of signal processing systems with passive failure detection

In this section, the integrity function will be derived for an electronic signal processing system with passive failure detection for the system structure shown in fig. 4.1. Corresponding to the assumption that sufficient spare units are available to repair each failure which has been detected (section 3.), fig. 4.1 shows a single system channel consisting of the signal processor (SP) and the failure detector (PD). The PD consists of the comparator and a second signal processor which produces the signal for comparison. A detected failure will be repaired by switching over to a spare unit of the same structure. This structure is, for instance, similar to that of a duo-duplex system with passive failure detection. With regard to the following analysis this structure has significant differences to other system layouts, where passive failure detection is carried out by majority- or mid-value voting etc. For these latter systems, more aspects have to be accounted for than is outlined in the following sections. In particular, the event of double- or multi-failures becomes much more important. This will briefly be covered in the appendix.

For a system structure corresponding to fig. 4.1 two aspects in addition to redundancy constraints are important for system integrity. These are the effects of

- a failure in the failure detector and
- dormant failures in the signal processor.

Both will be covered in the following.

#### 4.1 A failure in the failure detector (PD)

So far as is known, there exists no system with passive failure detection which can thoroughly diagnose failures in the failure detector itself. This kind of failure mode is very undesirable. A subsequent failure within the signal processor cannot be prevented from propagating into other connected units because the failure detector failed to alarm. Therefore, this case has to be covered by the integrity function.

For the sake of the following analysis two main failure modes are defined:

1. SPF - failure within the active signal processor (SP)
2. PDF - failure within the failure detector.

The second is the one to be mainly treated in this section.

The failure PDF is separated further into two categories with

PDF<sub>+</sub> designating a failure of the failure detector which will be indicated and  
PDF<sub>-</sub> designating a failure of the failure detector which will not be indicated.

The statistic characteristics of these failure modes are of the same nature as given for electronic components in equ. (3.1) and (3.2). The density functions are:

$$(4.1) \quad f_{PD} = \lambda_{PD} \cdot e^{-\lambda_{PD}t}; \quad f_{SP} = \lambda_{SP} \cdot e^{-\lambda_{SP}t}$$

$$f_{PD+} = \lambda_{PD+} \cdot e^{-\lambda_{PD+}t}; \quad f_{PD-} = \lambda_{PD-} \cdot e^{-\lambda_{PD-}t}$$

For the failure event PDF<sub>-</sub>, the integrity function must at least include the probability that prior to a possible failure SPF a failure PDF<sub>-</sub> has occurred. This event is designated by {C} and is considered to be equivalent to the loss of the system integrity. The event {C} can be illustrated by the diagram in fig. 4.2 using t<sub>SP</sub> and t<sub>PD-</sub>, the time values of the failure events, as the random variable coordinates. All possible combinations of random events SPF and PDF<sub>-</sub> within the elapsed time of operation are contained in the shaded area. {C} is dependent on the actual time of operation t<sub>a</sub> and it attains its maximum, when t<sub>a</sub> approaches t<sub>e</sub>, the end of the mission. The integrity function, which only considers this failure event, is designated as R<sub>PD-</sub>(t<sub>a</sub>). It represents the probability that {C} does not occur within the time interval (0, t<sub>a</sub>):

$$(4.2) \quad R_{PD-}(t_a) = 1 - F_{PD-}(t_a) = 1 - P\{C\}$$

The probability P {C} can be determined from fig. 4.2 by integrating the joint density function of the events SPF and PDF<sub>-</sub> over the shaded area (ref. 5):

$$(4.3) \quad P\{C\} = \int_0^{t_a} \int_0^{t_{SP}} f_{SP} \cdot f_{PD-} \cdot dt_{PD-} \cdot dt_{SP}$$

The events SPF and PDF<sub>-</sub> are independent. As was done in equation 4.3 the product of the density functions of SPF and PDF<sub>-</sub> can be substituted for the joint density function. The densities are given by

$$(4.4) \quad f_{SP}(t_{SP}) = \lambda_{SP} \cdot e^{-\lambda_{SP}t_{SP}} \quad \text{and}$$

$$f_{PD-}(t_{PD-}) = \lambda_{PD-} \cdot e^{-\lambda_{PD-}t_{PD-}}, \text{ such that the integration in equ. (4.3) results in}$$

$$(4.5) \quad P\{C\} = 1 - e^{-\lambda_{SP}t_a} - \frac{\lambda_{SP}}{\lambda_{SP} + \lambda_{PD-}} (1 - e^{-(\lambda_{SP} + \lambda_{PD-})t_a})$$

In fig. 4.3, (1 - R<sub>PD-</sub>(t<sub>a</sub>)), which is equal to P {C}, is plotted for different values of the rate of the failure event PDF<sub>-</sub>. The figure shows that for a failure rate  $\lambda_{SP} = 10^{-4}/[h]$ , a realistic magnitude, the integrity function decreases significantly from 1.

#### 4.2 Dormant failures

As opposed to analog systems, the information in digital systems is split up bit-wise and the processing functions also are often separated component-wise so that there are only two signal states (0, 1). During operational modes a great number of components are temporarily not actively involved in the process. During this period of idle or stationary operation these components cannot be tested by passive failure detection

and the corresponding failures are known as dormant failures. This kind of failures, of course, has a great effect on the integrity function. It follows that the integrity function not only depends on the statistical distribution of component failures but also on the distribution of the excitation of each system component and the corresponding operational mode. For the case of an unfavourable statistical distribution of the operational modes, i.e. a great probability of long time intervals between failure occurrence and failure detection, the system integrity may be lost a long time with no indication. By the way, a special case of a dormant failure which will not be indicated at all is the PDF- as defined in section 4.1. Only extensive software, which will not be dealt with in this article, can bring some relief to this problem. In this section, the effect of dormant failures on the integrity function will be described.

In a similar manner to what was done in section 4.1, the set of potential dormant failures within the signal processing unit will be defined here by the set  $\{D\}$ . Then, the integrity function can be written as

$$(4.6) \quad R^*(t_a) = 1 - F^*(t_a) = 1 - P^*\{D\}$$

or, including both the signal processor and the failure detector, the resulting integrity function is

$$(4.7) \quad R(t_a) = 1 - F(t_a) = 1 - P\{C + D\}$$

Fig. 4.4 shows the area of all elements of  $\{D\}$  for a signal processing component  $k$  in the  $t_{SP}(k), \Delta t_{MOD}(k)$  plane. The random variable  $t_{SP}(k)$  gives the time of the failure event in component  $k$ , and  $\Delta t_{MOD}(k)$  represents the time interval between the failure in component  $k$  and the next attempt to operate in mode  $k$  addressed to component  $k$ . The shaded area in fig. 4.4 represents the subset  $\{D(k)\}$  for any  $k = 1, 2, \dots, N$  of the complete set  $\{D\}$ . The actual time of operation  $t_a$  and the mission completion time  $t_e$  define the limits on the area  $\{D(k)\}$ . If the turn on rate for the operational mode  $k$ , designated, as  $\gamma_{MOD}(k)$ , is assumed constant, then the density function for the next turn on can be considered to be of the exponential type. Therefore, in the course of this article, it will be assumed that

$$(4.8) \quad f_{\Delta t_{MOD}(k)} = \gamma_{MOD}(k) e^{-\gamma_{MOD}(k) \cdot t}$$

A more precise formulation of the density function can be gained from experimental data. Assuming that the events of operational mode turn on and of failures are independent, the probability of event  $\{D(k)\}$  is then obtained by

$$(4.9) \quad P\{D(k)\} = \int_0^{t_a} \int_0^{t_e - t_{SP}(k)} f_{SP}(k) \cdot f_{\Delta t_{MOD}(k)} d\Delta t_{MOD}(k) dt_{SP}(k) \\ - \int_0^{t_a} \int_0^{t_a - t_{SP}(k)} f_{SP}(k) \cdot f_{\Delta t_{MOD}(k)} d\Delta t_{MOD}(k) dt_{SP}(k)$$

By substituting for the density functions corresponding to equs. (4.1) and (4.8) and evaluating the integrals,

$$(4.10) \quad P\{D(k)\} = \frac{\lambda_{SP}(k)}{\lambda_{SP}(k) - \gamma_{MOD}(k)} \cdot (1 - e^{-t_a(\lambda_{SP}(k) - \gamma_{MOD}(k))}) \cdot (e^{-t_a \gamma_{MOD}(k)} - e^{-t_e \gamma_{MOD}(k)})$$

Since the complete set  $\{D\}$  has to be considered for the integrity function in equs. (4.6) and (4.7),

$$(4.11) \quad \{D\} = \{D(1)\} + \{D(2)\} + \dots + \{D(N)\}$$

the probability of the event  $\{D\}$  can be derived from the probabilities of the events  $\{D(k)\}$ , corresponding to equ. (4.10), by

$$(4.12) \quad P\{D\} = 1 - \prod_{k=1}^N (1 - P\{D(k)\})$$

For equ. (4.12) it is assumed for the sake of simplicity that the events  $\{D(k)\}$  are independent. This may not be true in real cases. Most digital systems are coupled in a way that this assumption does not hold. Then the probability  $P\{D\}$  has to be derived by the sum of the probabilities  $P\{D(k)\}$  and their coupled terms. Higher order couplings can be neglected because of their insignificant contribution to the result of  $P\{D\}$ . For example, if only pairs of coupling terms are considered, the relationship for  $P\{D\}$  is given



by

$$(4.13) \quad P\{\underline{D}\} = \sum_{k=1}^N P\{\underline{D}(k)\} - \sum_{l=1}^N \sum_{k=1}^N P\{\underline{D}(k), \underline{D}(l)\} \quad k > l$$

To determine the coupling terms  $P\{\underline{D}(k), \underline{D}(l)\}$ , the fact that more than one operational mode is addressed to each component  $k$  and that each operational mode may be addressed to several components must be taken into account. Although, this can also be considered analytically in rather simple terms by use of a combinational matrix, no further outline will be given in this article.

In the simplest case, assuming independence; equ. 4.7 can be reformulated to yield

$$(4.14) \quad R(t_a) = 1 - (P\{\underline{C}\} + P\{\underline{D}\} - P\{\underline{C}, \underline{D}\}) \\ = 1 - [P\{\underline{C}\} + P\{\underline{D}\} (1 - P\{\underline{C}\})]$$

Further, substituting for  $P\{\underline{D}\}$  using equ. 4.12 yields

$$(4.15) \quad R(t_a) = 1 - [P\{\underline{C}\} + [1 - \prod_{k=1}^N (1 - P\{\underline{D}(k)\})] (1 - P\{\underline{C}\})]$$

Fig. 4.5 shows a diagram, where equ. (4.15) is evaluated for a mission length of  $t_e = 1$  [h], a typical hardware example of a signal processing unit with

$$\lambda_{SP} = 10^{-4}/[h] \\ \lambda_{SP(k)} = 10^{-5}/[h] \\ \lambda_{PD} = 0,5 \cdot 10^{-4}/[h]$$

and a mode configuration of  $N = 10$  or  $100$ . The rate of mode turn-on  $\gamma_{MOD(k)}$  is assumed to be the same for all modes. It varies from  $0,1/[h]$  to  $10/[h]$ . As can be seen from the plot, the more modes in which the signal processor are operating, the lower the integrity levels. The rate of mode turn-on also changes the time behavior of the integrity function. For a rate of 1 turn-on per 1/10 of an hour, the integrity function has a very shallow minimum at a rather small value of the mission time. For a rate of 1 turn-on per 10 hours the integrity function decreases continuously up to about 90 per cent of the complete mission time  $t_e$ . Of course, at the end of the mission, the integrity function recovers to the value of  $(1 - P\{\underline{C}\})$ , because, by definition, for  $t_a = t_e$  the value of  $P\{\underline{C}\}$  becomes zero.

#### 4.3 Constrained redundancy

In the foregoing sections the assumption was made that *detected* failures are repaired by available spare units right away no matter how many failures have occurred. This was done to eliminate the effect of finite redundancy on the integrity and to focus only on the effect of the performance of the failure detection. Thus, in the foregoing sections, detected failures were without influence on the integrity function.

However, if the redundancy is constrained the system integrity is also influenced by detected failure events because of the resulting reduction in available redundancy. For a signal processor with as many as  $x$  redundant units as shown in fig. 4.1, the integrity function, i.e. the probability of no failure occurrence in all of the  $x$  units until the actual time of operation  $t_a$ , is

$$(4.16) \quad R(t_a, x) = x \cdot R(t_a) = 1 - F(t_a)^x$$

Substituting for  $R(t_a)$  from equ. (4.7)

$$(4.17) \quad R(t_a, x) = x \cdot [1 - P\{\underline{C} + \underline{D}\}]$$

Certainly, considering the criterion for a mission abort, which for instance should ensure a safe return, the probability of surviving the mission or its complement, the probability of total loss within the mission, becomes important. In that case, using the results of the foregoing sections, the survival of the mission is to be excluded, if for a system with  $Z$  units still without detected failures, out of  $x$

1. a failure of the type  $PDF_{-}$  occurs in one of the  $Z$  units
2. all  $Z$  remaining units have failed.

The second condition includes all combinations of  $Z$  unit failures corresponding to the pattern:  $i$  undetected failures at the actual time  $t_a$  and  $Z - i$  failures within the time interval  $(t_a, t_e)$ .

Thus the probability of total loss within the mission as a function of the actual time of operation can be formulated as

$$(4.18) \quad F_{TL}(t_a) = 1 - R_{TL}(t_a)$$

$$1 - R_{PD}(t_a) + \sum_{i=0}^Z \binom{Z}{i} R(t_a)^{Z-i} (1 - R^*(t_a))^i \cdot (1 - R^*(t_a, t_e))^{Z-i}$$

This is applicable for a criterion for the mission abort of, for instance,

$$(4.19) \quad F_{TL}(t_a) \leq F_{TL_{MAX}}(t_a)$$

$R^*(t_a)$  from equation (4.6) is substituted into equation (4.18) and  $R(t_a, t_e)$  can be derived in a manner similar to that used for equ. (3.3). The expression  $(1 - R_{PD}(t_a))$  represents the system loss due to a failure PDF- within the failure detector.

In fig. 5.3, equ. (4.18) is evaluated for the numerical example of fig. 4.5. Remarks are given in the next section in conjunction with corresponding results for active failure detection.

#### 5. The integrity function of signal processing systems with active failure detection

As defined in section 2, active failure detection is in a prearranged way, such that a thorough check of all components is made within deterministic time intervals. The test procedure can be either sequential or simultaneous (ref. 3, 4). No realization of a continuous active failure detector with no delay between failure occurrence and failure detection is known, but the test cycle can be chosen as small as necessary so that the probability of failures known as double failures is small enough to be neglected. Double failures disable the comparison check because there are similar failures in both the signal processor and the signal used for the comparator, occurring within the time interval without test activity.

For the system with active failure detection, the same general system structure as shown in fig. 4.1 for the system with passive failure detection can generally be used.

Three distinctions should be noted:

1. Inherently several comparators are required within the single channel structure of a signal processing system with active failure detection.
2. A selfdiagnosing comparator as one crucial part of the active failure detector is feasible (ref. 3) as contrasted with the passive failure detection case.
3. In most applications the complete signal processor need not be doubled in order to generate the signals to be compared to each other.

Therefore, by virtue of the selfdiagnosing comparators and the complete test within small time intervals the integrity function of a signal processor with active failure detection is not dependent on the effects of dormant failures. Thus, if the test cycle time is small enough, about the only effect on the integrity to be considered comes from the redundancy constraints. In order to demonstrate this effect, the integrity function will be again derived for the case of unconstrained redundancy.

##### 5.1 Without redundancy constraint

If no redundancy constraints are assumed, the same assumptions that were used in sections 4.1 and 4.2 are valid, i.e. all failed components are repaired in the same instant, when their failure is detected. The test cycle time of the selfdiagnosing active failure detector (SAFD) is chosen as  $T$ , such that the integrity function for the values of the actual time  $t_a$

$$(5.1) \quad t_a = v \cdot T, \quad v = 0, 1, 2, \dots, \text{ is}$$

$$(5.2) \quad R(t_a) = 1, \quad \text{i.e. all test information is available}$$

and the integrity is established. During the time intervals between the discrete test instants no information is given of interim failure events by the failure detection mechanism. When the actual time of operation  $t_a$  coincides with the  $v$ -th test cycle, such that  $vT \leq t_a \leq (v+1)T$  (fig. 5.1), then the probability of no failure event within the interval  $(vT, t_a)$  is identical to the integrity function. By virtue of the selfdiagnosing property, the effect of a failure in the failure detector is not different from that of a failure in the signal processor itself. As opposed to passive failure detection, only the total failure rate of both signal processor and SAFD components, designated as  $\lambda_{SP/AD}$ , is used. Thus (fig. 5.1), the integrity function is

$$(5.3) \quad R(t_a) = R(vT, t_a) e^{-\lambda_{SP/AD} (t_a - vT)} \quad \text{with } v = 0, 1, 2 \dots \text{ and } vT \leq t_a \leq (v+1)T.$$

Fig. 5.2 shows a plot, where equ. (5.3) is evaluated for the minimum value of  $R(t_a)$ , several values of  $\lambda_{SP/AD}$  and varying  $T$ . The shaded area in fig. 5.2 indicates the region where the  $T$ -values of known systems with SAFD have been chosen. It becomes obvious that the integrity function can be kept extremely small within the intervals between the test instants. The values of  $\lambda_{SP/AD}$  are chosen so that these results for the integrity function can be compared with those for the signal processor with passive failure detection in fig. 4.5.

## 5.2 Constrained Redundancy

In a similar way to that discussed in section 4.3, the influence of limited redundancy on the system integrity is also outlined for a system with selfdiagnosing active failure detection (SAFD).

The integrity function of the complete system is also determined by the integrity function of a single channel (equ. (5.3)) multiplied by the full number of units available for operation. As outlined in section 4.3, the probability of total system loss, which is also dependent on the integrity function, becomes more important in the context of the criterion for the mission abort. This probability can be given for the system with SAFD in a similar form to that outlined in section 4.3. Although the influence of the noncontinuous test information is negligible in most cases, it will be considered for the sake of completeness. For a system with  $Z$  remaining units available,

$$(5.4) \quad F_{TL}(t_a) = 1 - R_{TL}(t_a) = \sum_{i=0}^Z \binom{Z}{i} R(t_a)^{Z-i} (1 - R(t_a))^i \cdot (1 - R(t_a, t_e))^{Z-i}.$$

Although the similarity to equ. (4.18) is obvious, the results are different because of the predominance of the last term  $(1 - R(t_a, t_e))^{Z-i}$ . In contrast to  $R(t_a)$  of equ. (4.18),  $R(t_a)$  in equ. (5.4) is very close to one and does not fall below its minimum values as plotted in fig. 5.2 (see also fig. 4.5). This characteristic can be concluded also from fig. 5.3, where both equ. (4.18) and equ. (5.4) are evaluated for the signal processing system with passive failure detection used as an example in section 4, and also for an equivalent example of a system with SAFD. The curves for the latter case, plotted as dashed lines, decrease considerably with increasing actual time of operation, whereas for the case with passive failure detection, the probability of total system loss does not change much throughout the complete mission. This indicates that for passive failure detection the criterion for the mission abort derived from the amount of detected failures makes some sense. For systems with active failure detection however, the probability of total system loss can recover considerably with increased time of operation.

For a certain  $F_{TLMAX} = \text{const.}$ , established as a criterion and plotted in fig. 5.3, a first failure prior to  $t_1$  in a system with three redundant units and a mission time of 10 hours results in a mission abort in both cases. However, a first failure within the time interval  $(t_1, t_2)$ , as indicated in fig. 5.3, only results in a mission abort for the passive failure detection case. The system with SAFD can still be operated without functional degradation. Notice, that  $(t_1, t_2)$  is about one third of the entire mission time.

## 6. HFB 320 experimental system

For the purpose of investigating the feasibility of selfdiagnosing active failure detection (SAFD) in all parts of a digital fly-by-wire flight control system, the DFVLR is conducting an experimental program on the HFB 320 aircraft, as shown in fig. 6.1. The functional subsystems and components installed in the aircraft are illustrated on the same figure. Besides manual control on the mechanical back-up control system, the functional layout of this system includes three fundamental modes of operation for the purpose of stabilization and flight path control:

- Full automatic control
- Semiautomatic control, i.e. flight path demand control by use of the side grip controller (see fig. 6.2)
- Manual fly-by-wire control.

In this section, a brief description of the control computer which is the signal processing part of the system will be given with particular emphasis on the integrity. Essentially a new computer had to be developed for the purpose of incorporating active failure detection methods into the computer design. The computer structure is shown in fig. 6.3 where all peripheral systems, such as experimental keyboard, tape reader, power supply etc. are not illustrated. The failure detection is mechanised by hardwiring methods, but for this experimental program no use is made of hardware integration methods for miniaturisation purposes.

Before more details are given about computer subsystems, a brief outline of the logic structure will be presented.

### 6.1 Logic Structure of the Control Computer

The control computer has to be a real time computer which is one of the main factors for establishing the characteristics of the logic structure. The input-output data flow with respect to the peripheral system, such as sensors, the instrumentation and the actuation systems, is autonomously managed by the interfaces. The interface memory is accessible by the main program which is also the case for internal read only and random access memory, i.e. a direct memory access is mechanised with respect to the input-output data.

The computer memory is byte-organised. It is divided into two parts, the data and the program memory. The address domain for the instructions for data transfer and arithmetic operations only includes the addresses of the data memory and those of the interface memories. The capacity of the data memory is 960 bytes, of which 192 bytes (64 data words) are read only and the remaining random access.

The program memory comprises 7231 bytes. The instructions stored in this memory are of variable word length, i.e. 2, 3 or 4 bytes, depending on the amount of information to be contained. The instruction reservoir consists of 7 instructions with respect to the memory, 3 transfer instructions, 8 jump instructions, and 64 instructions for other functions. Also the potential for subprogramming exists. The internal information transmission is brought about by a bus system consisting of four types of buses, the instruction bus (IN), the operand bus (OF) and the result bus (RE) as data buses to and from the arithmetic and logic unit and the control feedback bus (CF) for enabling the next program step. All buses are working serially and are failure detected by SAFD.

### 6.2 Main Redundancy and Failure Detection Concept

The computer is split up into subsystems, so-called moduls. There are five functionally different moduls, the ALU, memory, input interface, output interface, and the failure detection management system (fig. 6.3). Except for the last two, all moduls are duplicated. The redundancy of the output interface is achieved internally. Because of its selfdiagnosing property the failure management system need not be duplicated, except the switching units. Also the data links are duplicated.

Each modul output of those moduls which are duplicated is linked to the failure management system by a switching unit such that only one of the two moduls is providing the data which are to be fed into another pair of moduls. Because of these switching units the failure management system is able to completely eliminate a failed modul from the process. The failure message is generated within that modul where the failure occurred. Besides indicating the failures and switching to spare moduls, the failure management system ensures fail-safe behavior for the case when two moduls of the same kind have failed.

In summary, because of the redundancy concept, the system survives one arbitrary failure. The system also survives without degradation in its functional performance when, in addition to the first failure, another failure occurs in a different modul. If either part of one pair of moduls has failed, the fail-safe routine is employed.

### 6.3 Modul Failure Detection

In this section, two moduls are described in more detail as typical examples in order to give more insight into how active selfdiagnosing failure detection is realized.

#### 6.3.1 Arithmetic and Logic Unit (ALU)

As shown in fig. 6.4 via the operand bus from the memory of the input interface the data to be worked on are fed into the ALU modul. The output of the ALU is transmitted to the memory and the output interface via the result bus. The instruction bus and the control feedback bus couple the ALU to the main control unit within the memory modul. There are buffering registers because the transmission frequencies used on the buses are different from that internal the ALU. The accumulator works serially in order to keep the number of components down. The internal control unit is a read only memory which can easily be checked.

In order to detect failures actively within the accumulator two methods are applied. The first method is that of using Manchester coding to represent the information by signal changes (e.g. 1 to 0) instead of the signal amplitude. The occurrence of these signal changes is equivalent to a dynamic test signal and can be considered as such for the purpose of selfdiagnosing active failure detection (ref. 3). The data entering the registers are also coded in this way. The second method is the sequential test. If SAFD is to be realized for the ALU, for simplicity's sake, both methods are used for the testing procedure. For further reduction of the amount of components, the ALU works serially. In order to carry out the sequential test, the registers are lengthened for a test sequence. This sequence is coded in the same way as the main signal and is monitored at the ALU input and output. The test sequence can be chosen such that all possible internal states of the ALU components are tested. It has been shown that a test sequence of 8 bit is sufficient to achieve this complete test. When an addition is to be carried out, first the test sequence passes the adder. As soon as the test response is provided, i.e. after 8 shifting cycles, the addition itself can start.

#### 6.3.2 Input Interface

The input interface contains 64 data channels of both analog and digital inputs. Periodically all of this input information is fed into the interface memory in a predefined order. The failure detection with respect to the analog inputs, in particular for the multiplexer, has been achieved by a combination of detection methods. On one hand the analog multiplexer is doubled (fig. 6.5) and its outputs are compared by



a precision comparator. On the other hand an additional spare input channel is used for a test signal which is generated by the internal processing controller. A certain number of discrete amplitude levels are contained in the test signal such that the multiplexer is tested in its full amplitude. The comparison takes place within the processing controller after the signal has passed the A/D-converter. Again the digital signal is coded by use of the Manchester code. The integrity of the coding mechanism is checked at the memory input and output. For that purpose all received data are read after they are fed into the memory. The processing of the digital inputs is carried out in a similar but, of course simpler way.

## 7. Conclusions

Despite the fact, that there are a great number of system designs and developments for digital flight control systems, not enough work has been done in the area of quantitative comparison of these systems with respect to the integrity and the mission survival probability. This article contributes to the analytical use of the integrity function which contains the information about the system status at the actual time of system operation and which can be used for the determination of the probability of system loss during the remaining time of the mission.

In particular, the influence of the quality of the malfunction detector on the integrity function is evaluated for two versions, passive failure detection and active failure detection. It is shown that for passive failure detection the integrity function as a function of the actual time of operation is deteriorated considerably by possible events of dormant failures. Following from that, the probability of system loss during the rest of the mission as a function of the actual time of operation is increased significantly relative to the corresponding results for selfdiagnosing active failure detection (SAFD). For a system with SAFD, a great amount of extra operation time can possibly be gained before a mission abort becomes necessary.

Of course, rather simple systems have been compared within this article, but the analytical relationships are also valid in principle for quantitative comparisons of more complex systems.

## 8. Appendix

If the monitoring signal processor in fig. 4.1 is taken from the system by replacing its function by the spare redundant units, the system structure of a system with malfunction elimination by majority voting or mid value voting is obtained. In this case, in addition to that as described in section 4, a critical situation arises, if half of the amount of available redundant units have undetectedly failed by dormant failures.

The probability of  $Z/2$  dormant failures of the same kind in a system with  $Z$  available redundant units and no failures detected in these units until the actual time  $t_a$ , is as a function of  $t_a$  (equ. 4.10)

$$(A.1) \quad F_D(Z/2; t_a) = \sum_{k=1}^N \sum_{i_1=1}^Z \sum_{i_2=1}^Z \dots \sum_{i_{Z/2}=1}^Z P\{\underline{D}(k, i_1), \underline{D}(k, i_2), \dots, \underline{D}(k, i_{Z/2})\}$$

$$i_1 \neq i_m$$

The example of an available redundancy of four units would yield

$$(A.2) \quad F_D(2; t_a) = \sum_{k=1}^N \sum_{i_1=1}^4 \sum_{i_2=1}^4 P\{\underline{D}(k, i_1), \underline{D}(k, i_2)\}$$

$$i_2 > i_1$$

or by approximating for equ. (A.2), assuming that

$$\{\underline{D}(k, i_1)\} \text{ and } \{\underline{D}(k, i_2)\} \text{ are independent and that}$$

$$P\{\underline{D}(k, i_1)\} = P\{\underline{D}(k, i_2)\}$$

$$(A.3) \quad F_D(2; t_a) = 6 \cdot N [P\{\underline{D}(k)\}]^2$$

9. References

- 1 Design and Flight Experience with a Digital Fly-by-Wire Control System in a F-8 Airplane.  
*D.A. Deets, K.J. Szalai*, AGARD-CP-137, 1974
- 2 Definition Study for an Advanced Fighter Digital Control System,  
*McDonnell Aircraft Company*, AFFDL-TR-75-59
- 3 Digital Fly-by-Wire Control System with Selfdiagnosing Failure Detection,  
*R. Orken, H.P. Joenok, L. Tacke, M. Gottschlich*, AGARD-CP-137, 1974
- 4 Monolithisches Schaltkreissystem zum Aufbau von Fail-Safe-Schaltwerken,  
*H.-J. Lohmann*, Siemens-Bericht 2-2500-810, 1970
- 5 Probability, Random Variables and Stochastic Processes,  
*A. Papoulis*, Mc Graw Hill
- 6 Zuverlässigkeitsprobleme elektronischer Schaltungen,  
*W. Görke*, Bibliographisches Institut Mannheim, 1969
- 7 Mathematische Methoden der Zuverlässigkeitstheorie,  
*B.W. Guedenko, J.K. Beljajew, A.D. Solowjew*, Akademie-Verlag Berlin 1968

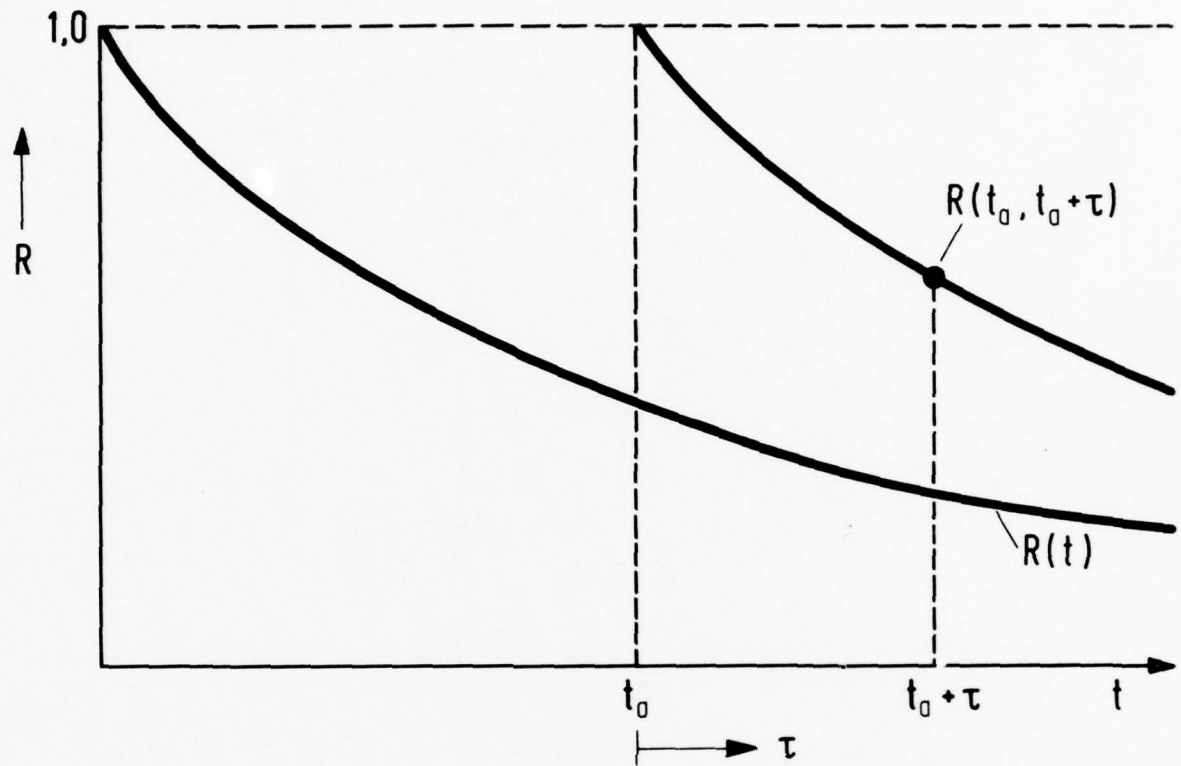


Fig. 3.1 RELIABILITY FUNCTION FOR THE INTERVAL  $(0, t)$  (NO FAILURE AT  $t=0$ ) AND THE INTERVAL  $(t_a, t_a + \tau)$  (NO FAILURE UNTIL  $t_a$ )

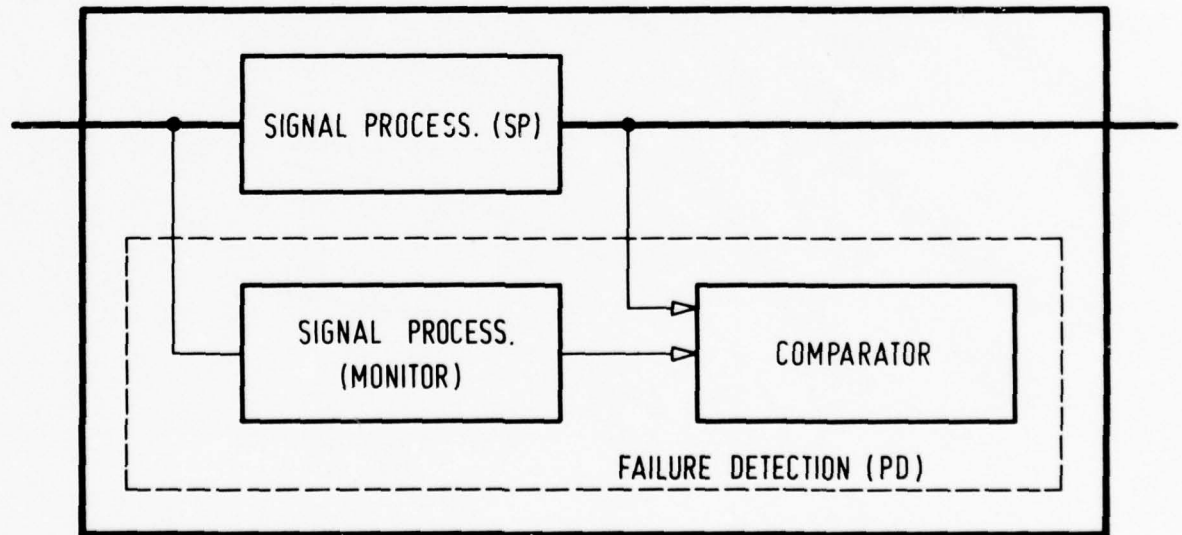


Fig. 4.1 SINGLE CHANNEL SYSTEM STRUCTURE WITH PASSIVE FAILURE DETECTION

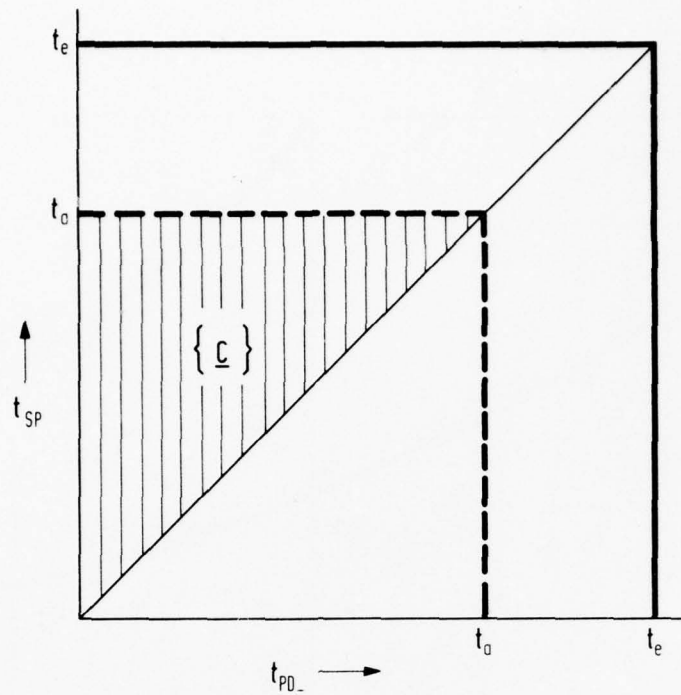


Fig. 4.2 AREA OF POSSIBLE EVENTS OF UNDETECTABLE FAILURES OF THE FAILURE DETECTION AS A FUNCTION OF THE ACTUAL TIME  $t_a$

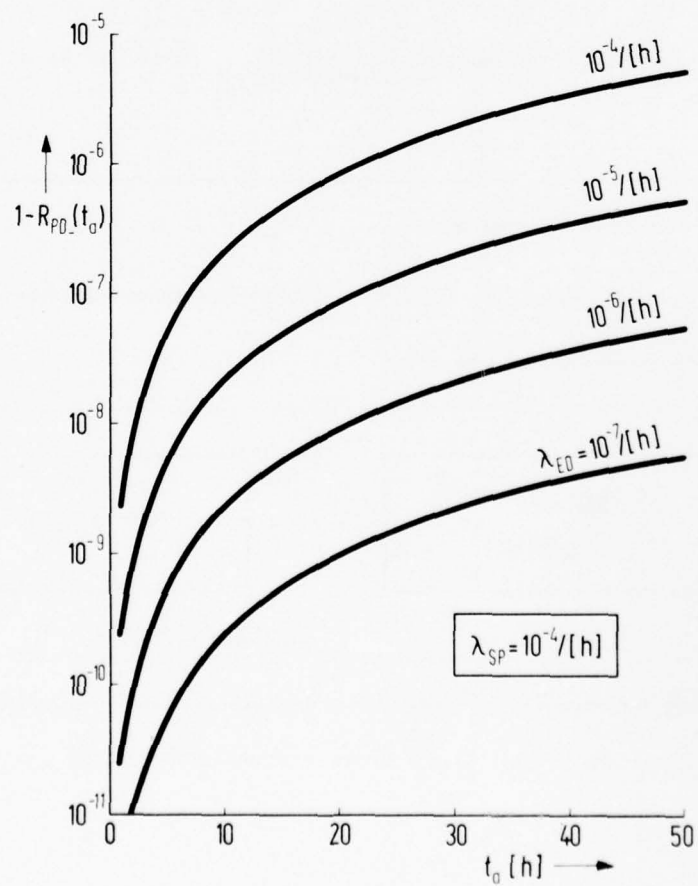


Fig. 4.3 PROBABILITY FOR UNDETECTABLE FAILURE OF FAILURE DETECTION AS A FUNCTION OF THE ACTUAL TIME  $t_a$



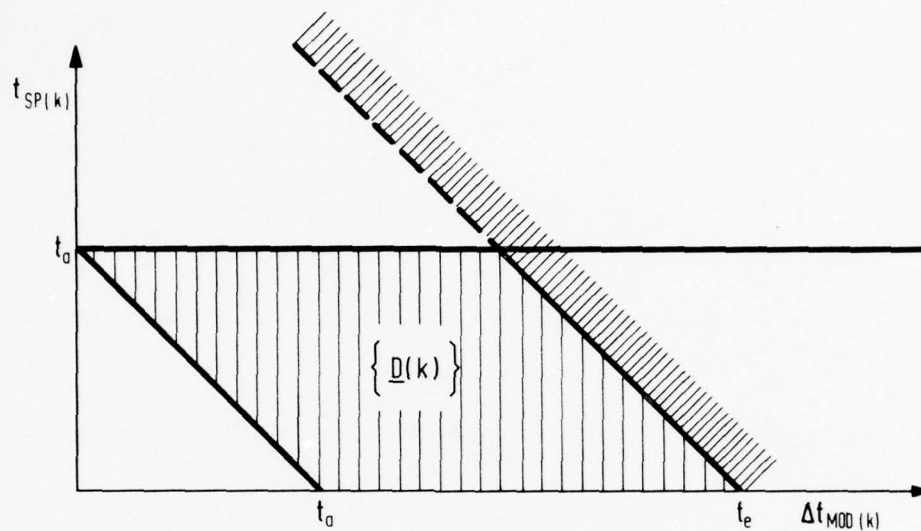


Fig. 4.4 AREA OF POSSIBLE EVENTS OF DORMANT FAILURES AS A FUNCTION OF THE ACTUAL TIME  $t_a$

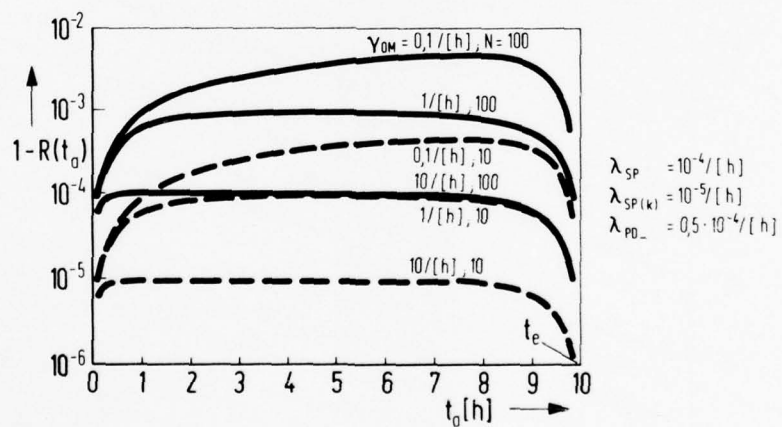


Fig. 4.5 INTEGRITY FUNCTION FOR SYSTEM CHANNEL WITH PASSIVE FAILURE DETECTION

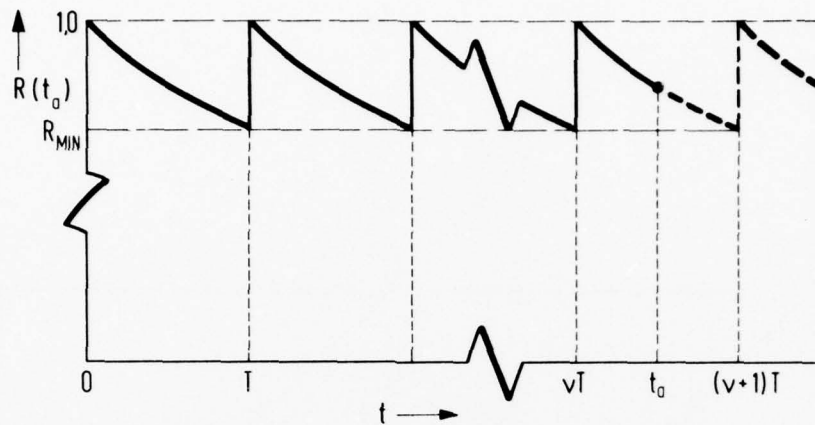


Fig. 5.1 INTEGRITY FUNCTION FOR SYSTEM CHANNEL WITH SELF-DIAGNOSING ACTIVE FAILURE DETECTION (SAFD)

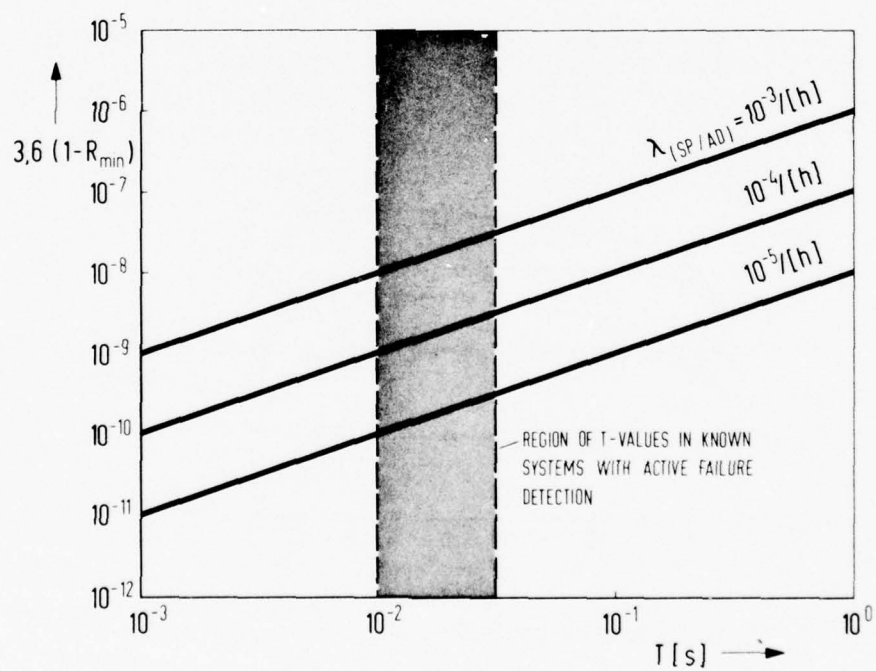


Fig. 5.2 MINIMUM VALUE OF THE INTEGRITY FUNCTION AS A FUNCTION OF THE TEST CYCLE TIME  $T$

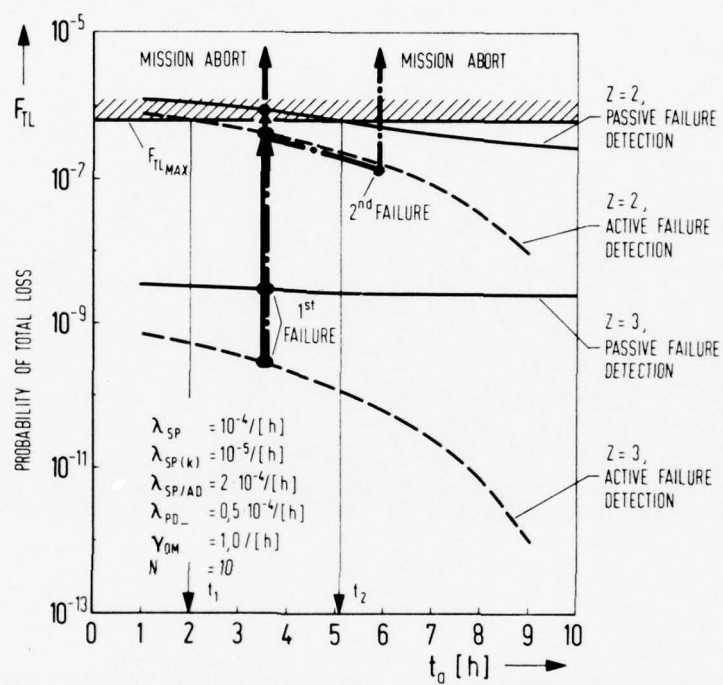


Fig. 5.3 PROBABILITY FOR TOTAL LOSS WITHIN THE MISSION TIME AS A FUNCTION OF THE ACTUAL TIME  $t_a$

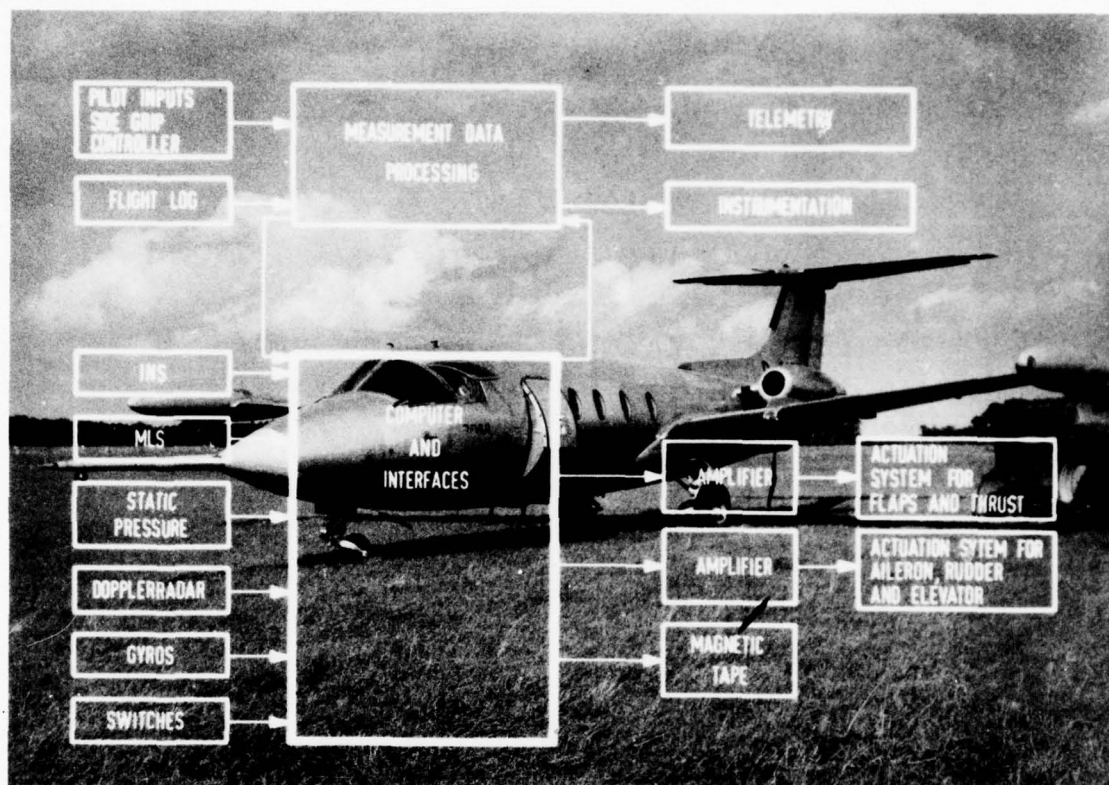


Fig. 6.1 HFB 320 EXPERIMENTAL SYSTEM



Fig. 6.2 HFB 320 SIDE GRIP CONTROLLER



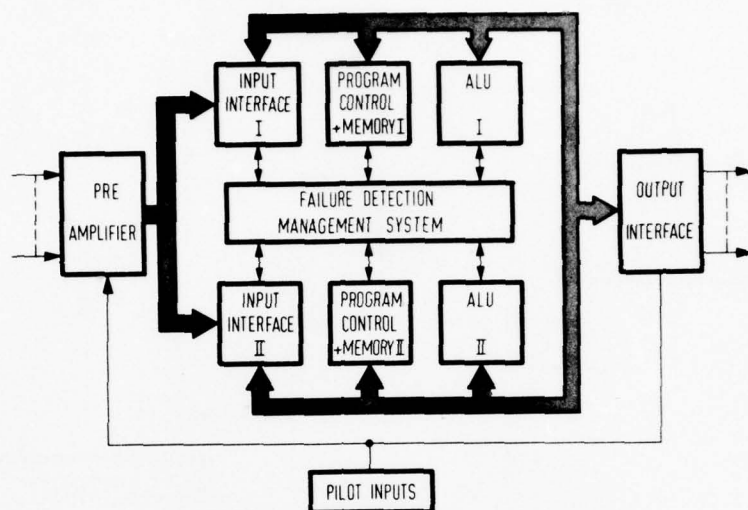


Fig. 6.3 STRUCTURE OF FLIGHT CONTROL COMPUTER WITH SAFD

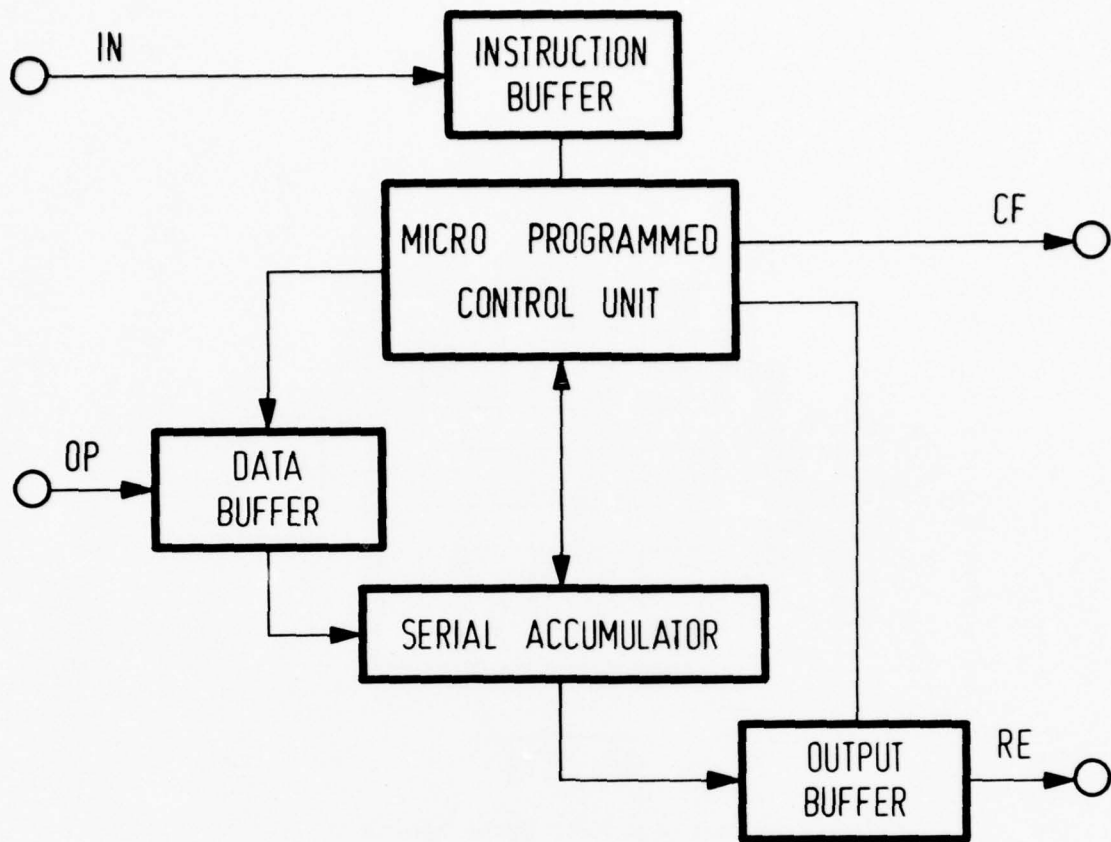


Fig. 6.4 MODUL STRUCTURE OF THE ARITHMETIC AND LOGIC UNIT

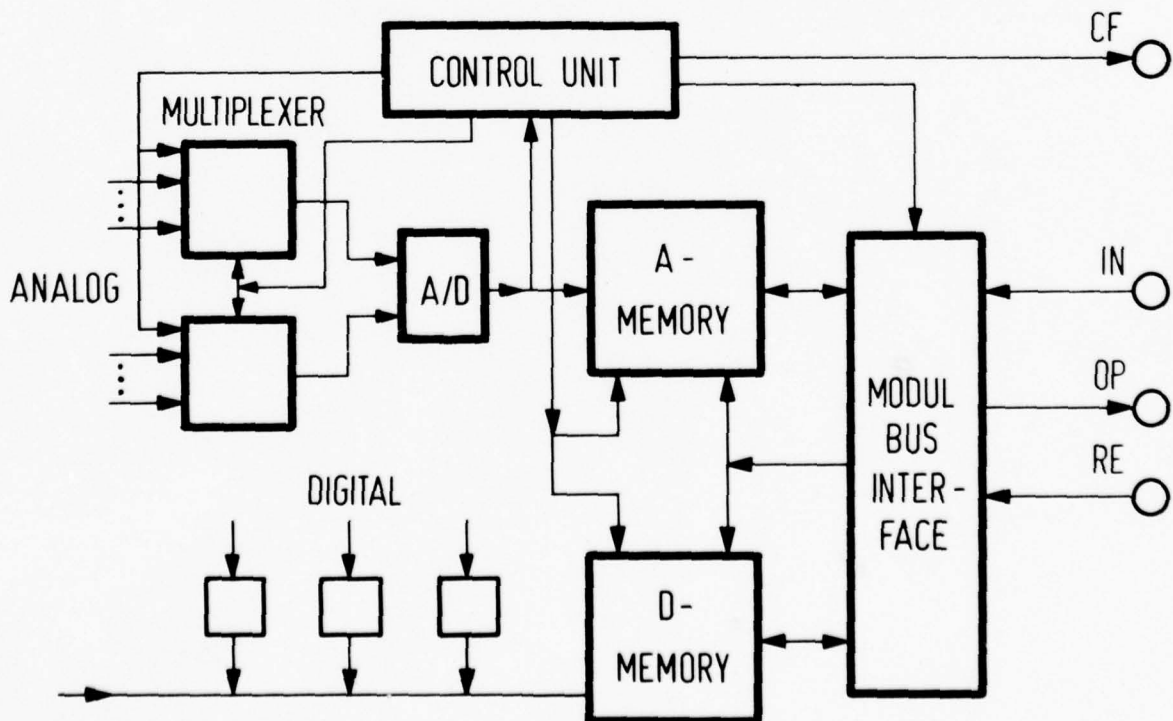


Fig. 6.5 MODUL STRUCTURE OF THE INPUT INTERFACE

# FAILURE SELF-DETECTION IN DIGITAL FLIGHT GUIDANCE SYSTEMS

by

H. Drtil

W. Meyer

Bodenseewerk Gerätetechnik GmbH

D-7770 Überlingen, Postfach 1120

## Summary

Flight guidance systems have to meet severe reliability requirements for reasons of flight safety. This applies in particular to aircraft equipment and flight missions that can no longer be controlled manually by the pilot. Digital realization of future flight guidance systems is expected for cost reasons. This technology facilitates the introduction of advanced redundancy procedures that take advantage of the characteristics of digital systems to self-detect failures through suitable procedures.

Procedures for detecting failures in the hardware of the signal processing have been developed. This failure self-detection is carried out by means of suitable test programmes, and it is supervised by an external supervisor. The design of this supervisor is based on the principle of a watch-dog-timer.

Two supervisor systems have been developed on the basis of this principle. The improved version allows in addition the execution of nominal/actual value comparisons in the supervisor, and this increases the failure self-detection probability.

The possible applications of the failure self-detection are discussed.

## 1. Introduction

The requirements to be met to ensure the reliability of flight guidance systems are very high for reasons of flight safety. This applies in particular to flight equipment and missions that can no longer be manually controlled by the pilot such as automatic landings under bad weather conditions or unstable aircraft (CCV).

The presently existing flight guidance systems have been designed in accordance with the principles of parallel redundancy. This resulted in a great amount of hardware and problems for testing and maintaining the systems.

The following characteristics will be typical for the next generation of flight guidance systems:

- increased performance requirements for the flight guidance
- economy requirements  
i.e. reduction to a minimum of the hardware
- improvement of the testability and maintainability
- reliability

The additional condition of economy can be met, for instance, by means of new non-conventional reliability principles.

Cost investigations show that with increasing complexity of the flight guidance systems, the price/performance ratio of digital systems is more favourable than for analog systems. Digital signal processing facilitates the introduction of advanced redundancy techniques. The characteristic of digital systems to automatically detect failures by suitable means can be utilized in particular.

The redundancy principles developed by Bodenseewerk consist in detecting and localizing hardware failures of the flight guidance system by means of suitable checking and test programmes.

The essential elements of a flight guidance system are:

- signal processing unit composed of
  - Processor (CPU)
  - Read-only memory (ROM)

- Working storage (RAM)
- Interface
- Real-time clock
- Sensors
- Actuator system
- Pilot's control panel

The failure self-detection procedure allows failures to be detected and localized in the signal processing unit.

The monitoring of sensors, actuator system and pilot's control panel can be ensured by means of suitable software monitors integrated in the signal processing unit.

## 2. Principles of Failure Self-detection

The principles of the failure self-detection are described in the following:

In self-monitored systems it has to be basically ensured that the self-monitoring is supervised by an external self-contained unit.

This external supervisor must be fail-safe within the scope of the specified failure rates. The supervising of the failure self-detection is necessary as the processor of the signal processing unit can only detect failures as long as it is functionally in order. The external supervisor assumes the evaluation and monitoring of the failure self-detection programmes and disconnects the monitored control channel (figure 1) in the event of a failure.

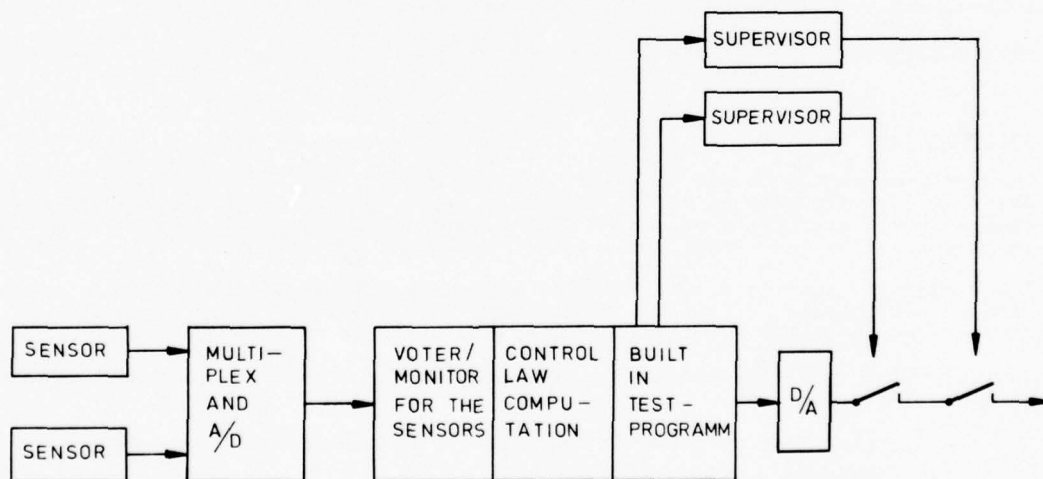


FIGURE 1  
FAILURE SELF-DETECTION IN AUTOMATIC FLIGHT GUIDANCE SYSTEMS

### 2.1 Automatic Supervisor System (ASS)

If only one monitoring is carried out, an automatic supervisor system can be built up which operates in principle as a watch dog timer. Unless a reset instruction is given in time to the supervisor by the processor, a disconnecting instruction (figure 2) is carried out automatically by the supervisor after an adjustable delay



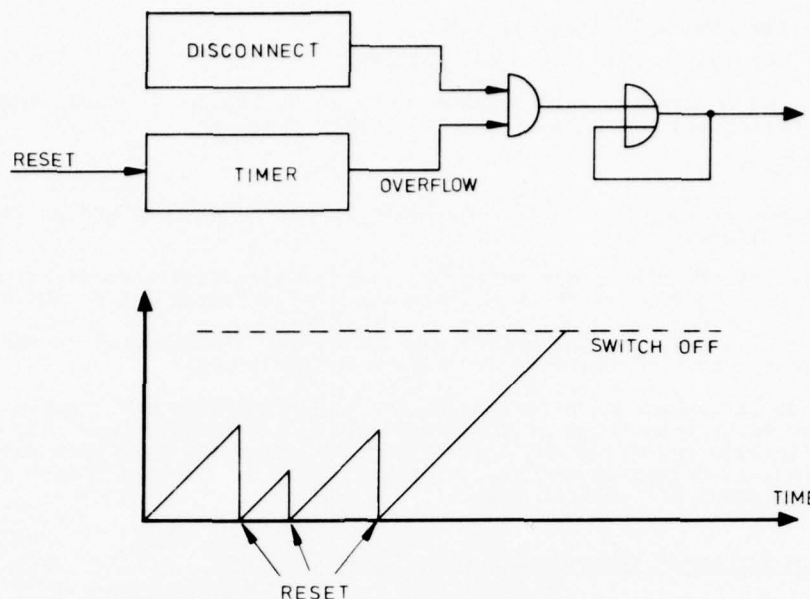


FIGURE 2  
AUTOMATIC SUPERVISOR SYSTEM (ASS)

This simple automatic supervisor detects and monitors therefore specific failures in the programme (e.g. infinite loops) as well as clock failures of the real-time clock and the clock generator.

In order to enable the supervisor to also detect failures in the power supply of the signal processing unit, it has been so designed that the shut-down or change-over relay assigned to the supervisor disconnects the channel in the event of a breakdown of the power supply.

Further failures in the signal processing unit can be detected by means of suitable failure self-detecting programmes. These programmes are worked out in flight during the intervals available between the computing cycles.

The following tests are carried out within the scope of this failure self-detection programmes:

- CPU test

A complete check of the instruction repertory is performed. A suitable test programme is worked off by presetting constant input values. By utilizing all instructions available in the processor, results are obtained which allow, by comparison with known nominal values, conclusions about failures in the processor hardware, to be drawn.

- ROM test

The contents of the read-only memory (ROM) are verified by parity-check in the columns.

- RAM test

The working storage (RAM) has to be checked in a different manner than the ROM since the contents of the RAM change continuously and are therefore not known.

The following items are checked by means of the developed software procedure:

- the memory cells
- the address registers
- the decoder network

- the read and write amplifier
- the switch-over units for read/write

By using respectively complementary input values, a double check is carried out by means of extensive test routines.

#### - Interface test

The interface is supervised with the known wrap-around procedure by using a software monitor.

After completion of each of the described individual tests, a reset instruction is given to the supervisor, if no failure has been detected during the test.

In the event of a failure, the supervisor can be set directly, or in the simplest case indirectly, by the programme entering a defined stop.

The described interplay of failure self-detection programme and supervisor has been checked within the scope of laboratory tests and flight tests. All failures artificially generated and occurred in this connexion have been detected by a failure self-detection and the supervisor, and the control system has never been switched off without reason.

### 2.2 Programmable Automatic Supervisor System (PASS)

An essential feature of this failure self-detection procedure was substantially improved about two years ago.

It is easy to see that certain failures cannot be covered by the described failure detection principle.

It has been shown that many tests performed by failure self-detection procedure end in a nominal/actual value comparison. This comparison cannot be performed in the signal processing unit as any failure of the instructions or hardware necessary for the comparison could result in a misinterpretation which might release, in general, a reset instruction to the supervisor irrespective of the result of the nominal/actual value comparison.

If all nominal/actual value comparisons are performed externally and independently, a substantial increase of the procedure reliability is possible.

For this purpose the supervisor was extended to a programmable automatic supervisor system. This was achieved without a too great increase in the hardware (figure 3).

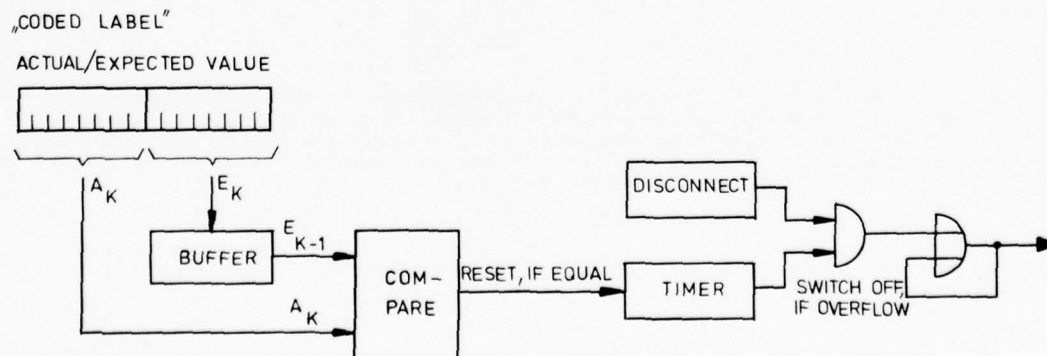


FIGURE 3  
PROGRAMMABLE AUTOMATIC SUPERVISOR SYSTEM (PASS)

Coded marks ( $A_k, E_k$ ) are offered to the PASS by the programme. These marks are continually varied by the coding procedure so that the more significant half of the mark is identical with the less significant half of the previously generated mark (figure 4).

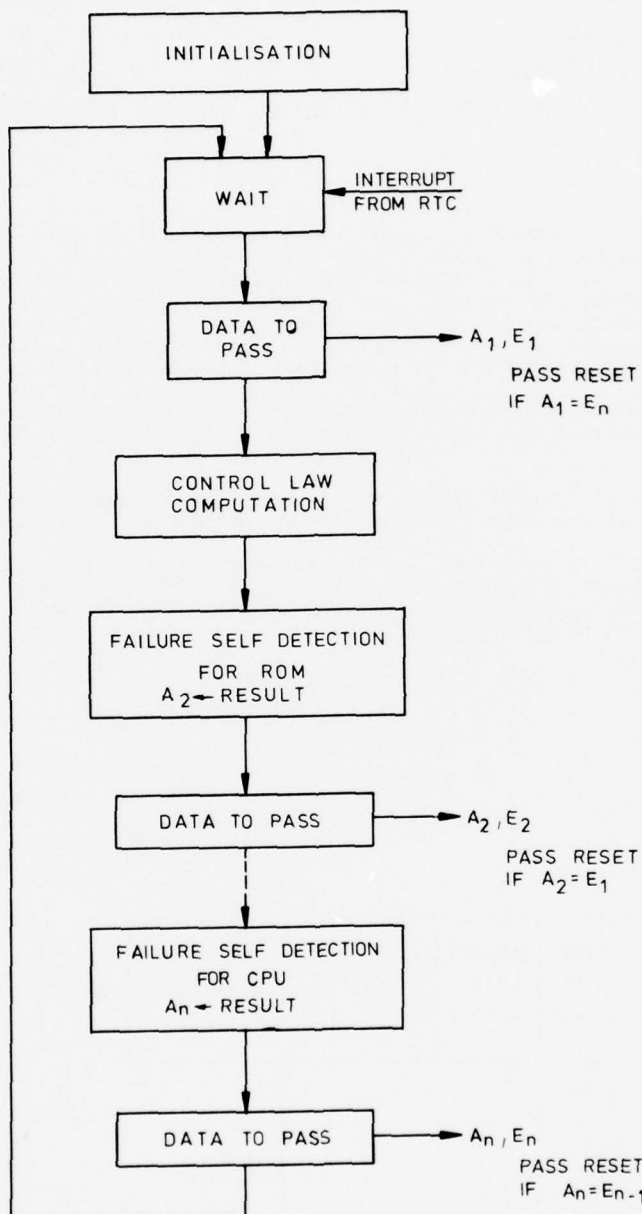


FIGURE 4  
INTERPLAY OF FAILURE SELF DETECTION AND PASS

The marks have therefore to be distributed in the programme so that a defined sequence is maintained during the running off of the programme. Due to the necessary equality of the more significant half of the mark and the less significant half of the previous mark ( $A_k = E_{k-1}$ ) actual/nominal value comparison independent of the computer can now be performed in the supervisor by transmitting to the supervisor the expected result  $E_{k-1}$  prior to each test. At the end of the test the actual result  $A_k$  of the test is transferred to the supervisor in the more significant half of the new mark. A control unit of the PASS compares the actual value  $A_k$  with the nominal value  $E_{k-1}$  contained in the buffer. In the event of

parity the control unit applies a reset signal to the timing element of the supervisor PASS. In the event of disparity a disconnect signal is generated in the supervisor.

This principle is generally used for the above described test programmes. In addition, the programme flow monitoring and hardware failure monitoring can be considerably improved concerning the following items by using the supervisor PASS:

- loop control
- jump instructions
- inadmissible frequency variations of the rel-time clock.

### 3. Applications

Taking into account the safety requirements to be met by a flight guidance system, the degree of system redundancy is derived from the flight-mechanical characteristics and the mission requirements the aircraft equipment has to meet. The selection of the redundancy concept, for instance parallel redundancy or failure self-detection, clearly depends on the respective specification, the weight, the volume and the electrical power consumption as well as on the latest state of the art and the associated costs for development, production and certification.

The certification costs can be very high for advanced technologies and procedures not yet proved in operational application such as the method for failure self-detection.

A conventional digital triplex system would therefore be used today for fail-operational applications, such as for instance, automatic bad-weather landing.

If it can, however, be proved that the described procedure of failure self-detection operates with a failure detection probability of  $1 - 10^{-4}$ , it would then be possible to use a duplex system with failure self-detection (figure 5) instead of a state-of-the-art triplex system.

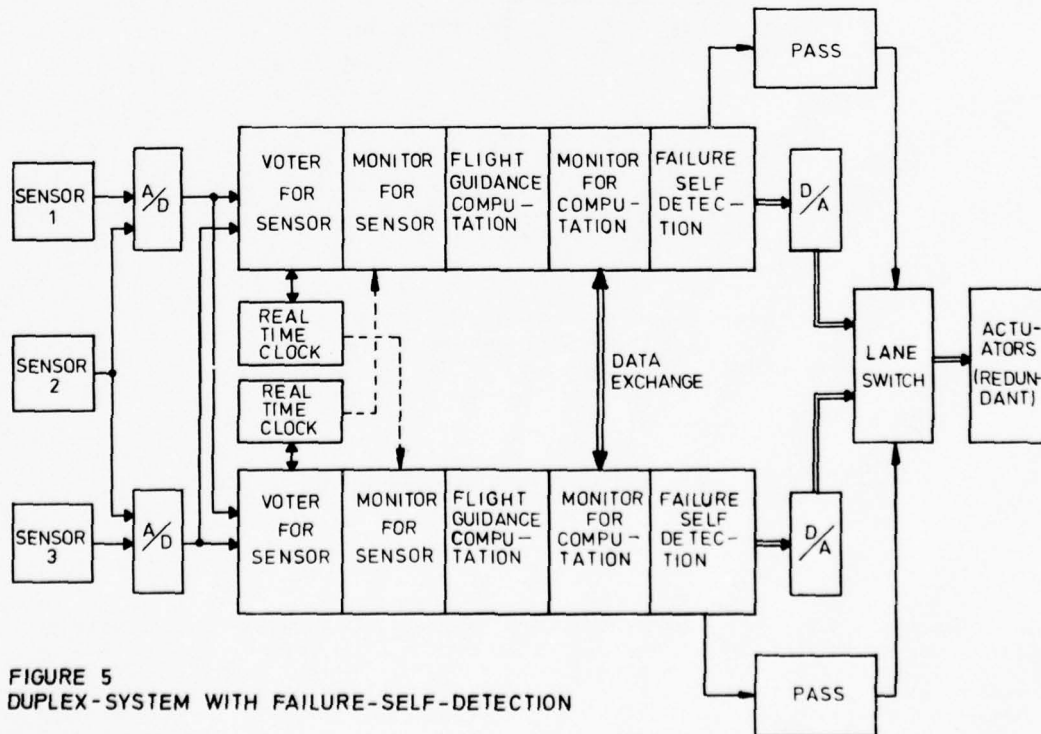


FIGURE 5  
DUPLEX-SYSTEM WITH FAILURE-SELF-DETECTION



The failure detection probability of  $1 - 10^{-4}$  has to be specified for fail-operational uses as a probability of about  $10^{-3}/h$  can be assumed for the hardware failure of one channel. Such a hardware failure is catastrophic if it is not detected by the failure self-detection .

The probability of catastrophic failure is therefore  $10^{-7}/h$  on the basis of the above mentioned assumptions. The failure self-detection is a promising procedure for the next aircraft equipment generation since the hardware amount is considerably lower than for conventional parallel redundant systems. The evidence of the failure detection necessary for the certification has to be furnished through future investigations.

SNEAK CIRCUIT ANALYSIS APPLICATION TO  
CONTROL SYSTEM DESIGN

Joe L. Wilson, Senior Engineer  
Robert C. Clardy, Sneak Circuit Analyst  
THE BOEING COMPANY  
P. O. Box 58747  
Houston, Texas 77058

## SUMMARY

An overview of the development and application of a circuit analysis technique will be presented in this paper. The technique is based on an aerospace discovery that topological criteria exist that can be used to recognize unplanned operational modes of a circuit. The analysis technique involves encoding circuitry data from detailed schematics for computer processing. The computer processing produces simplified, topological network trees which represent the system circuitry. The network trees are analyzed by the application of sneak circuit conditions. The results obtained from a variety of complex electrical systems analyses will also be presented as positive collaboration for this circuitry analysis technique.

## BACKGROUND

A sneak circuit is a latent path or condition in an electrical/electronic system which inhibits a desired condition or initiates an unintended or unwanted action. A sneak circuit is not caused by component failures, but is a condition that has been inadvertently designed into an electrical/electronic system. Sneak circuits often exist because many subsystem designers lack the overall system visibility required to electrically interface all subsystems properly. Some sneak circuits are evidenced as "glitches" or spurious operation modes and can occur in mature, thoroughly tested systems after long use. Sometimes sneaks are the real cause of problems blamed on electromagnetic interference or ground "bugs". Sneak Circuit Analysis should not, therefore, be confused with other unrelated analysis techniques such as:

- 0 Failure Modes and Effects Analysis
- 0 Fault Tree Analysis
- 0 Electromagnetic Compatibility Analysis
- 0 Parametric Analysis
- 0 Reliability Analysis
- 0 Hardware Oriented Testing and Troubleshooting

The unpredictable nature of sneak circuit conditions prompted the National Aeronautics and Space Administration (NASA) to fund an investigation to determine a method of identifying sneak circuit conditions before their occurrence could pose any possible threat to the safety of Apollo, Skylab or Apollo-Soyuz Test Program (ASTP) crew members.

The Boeing Aerospace Company began such a study in November 1967, which was composed of a detailed review of historical incidents of sneak circuit conditions in various electrical systems. A sneak circuit was defined for this study as "... a designed-in signal or current path which causes an unwanted function to occur or which inhibits a wanted function." The definition excluded component failures and electrostatic, electromagnetic, or leakage paths as causative factors. The definition also excluded improper system performance because of marginal parametric factors or slightly out-of-tolerance conditions.

The 1967 historical incident investigation resulted in two significant findings: (1) Sneak circuits are universal in complex electrical systems and their analogs, and (2) Topological criteria exist which enable recognition of all planned or unplanned operational modes within a system. Boeing developed a computer-aided analysis technique based on the 1967 findings which established the following analytical goals: (1) The simplification and reduction of electrical system detail schematics to topological network trees, (2) The recognition of the basic topographical patterns inherent in all circuitry, and (3) The application of the appropriate clues which have been found to characterize sneak circuit conditions.

## NETWORK TREE PRODUCTION

The first major consideration that must be satisfied to identify sneak circuit conditions is to insure that the data being used for the analysis represents the actual "as-built" circuitry of the system. Functional schematics, integrated schematics, and system level schematics do not always represent the actual constructed hardware. Detail manufacturing and installation schematics must be used because these drawings specify exactly what was built, contingent on quality control checks, tests, and inspections. However, manufacturing and installation schematics rarely show complete circuits. The schematics are laid out to facilitate hookup by technicians without regard to circuit or segment function. As a result, analysis from detail schematics is extremely difficult. So many details and unapparent continuities exist in these drawings that an analyst becomes entangled and lost in the maze. Yet, these schematics are the data that must be used if analytical results are to be based on true electrical continuity. The first task of the sneak analyst is, therefore, to convert this detailed, accurate information into a form useable for analytical work. The magnitude of data manipulation required for this conversion necessitates the use of computer automation.

Automation has been used in sneak circuit analysis since 1970 as the basic method of tree production from manufacturing-detail data. Computer programs have been developed to allow encoding of simplified continuities in discrete "from-to" segments from detail schematics and wire lists. The encoding can be accomplished without knowledge of circuit function. The computer connects associated points into paths and collects the paths into node sets. The node sets represent interconnected nodes that make up each

circuit. Plotter output of node sets and other reports are generated by the computer to enable the analyst to easily sketch accurate topological trees. The computer reports also provide complete indexing of every component and data point to its associated tree. This feature is especially useful in cross-indexing functionally related or interdependent trees, in incorporating changes, and in troubleshooting during operational support.

#### TOPOLOGICAL PATTERN IDENTIFICATION

Once the network trees have been produced, the next task of the analyst is to identify the basic topological patterns that appear in each tree. Five basic patterns exist: (1) The single line (no-node) topograph, (2) The ground dome, (3) The power dome, (4) The combination dome, and (5) The 'H' pattern. These patterns are illustrated in Figure 1. One of these patterns or several in combination will characterize the circuitry shown in any given network tree. Although at first glance a given circuit may appear more complex than these basic patterns, closer inspection reveals that the circuit is actually composed of these basic patterns in combination. As the sneak circuit analyst examines each node in the network tree, he must identify which pattern or patterns that node is part of and apply the basic clues that have been found to typify sneak circuits involving that particular pattern.

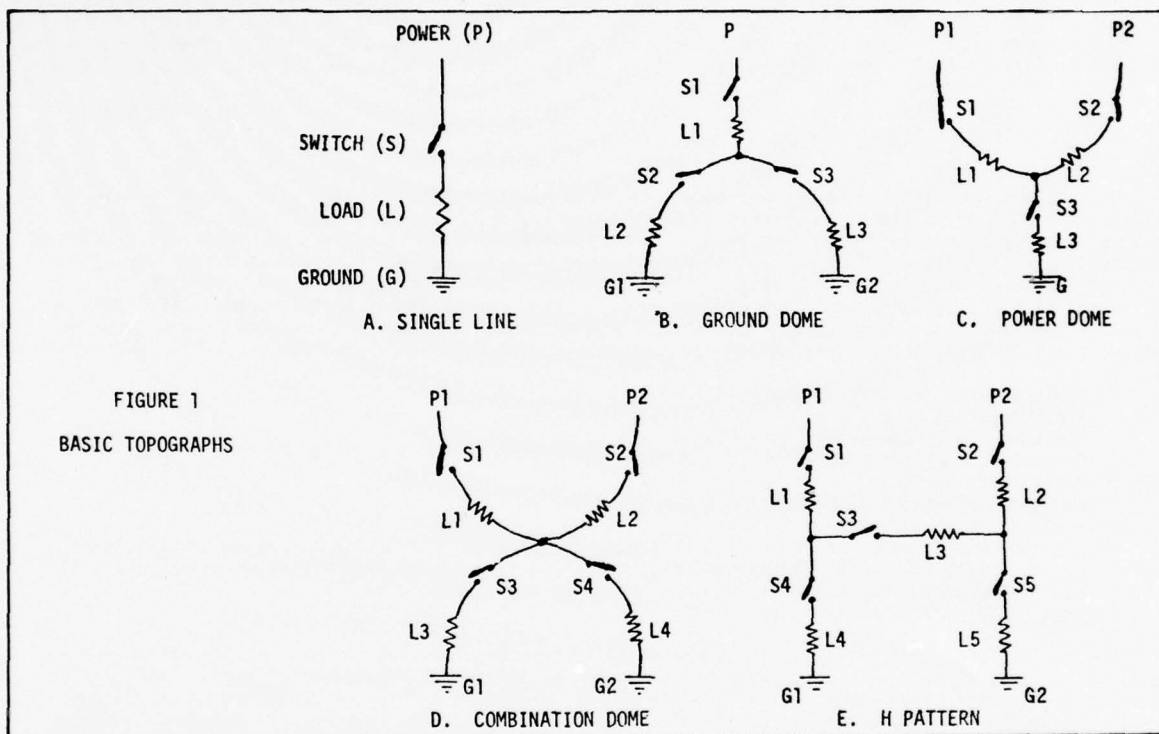


FIGURE 1  
BASIC TOPOGRAPHS

#### CLUE APPLICATION

Associated with each pattern is a list of clues to help the analyst identify sneak circuit conditions. These clues are questions that the analyst must ask about the circuitry in question. As an example, the single line topograph (Figure 1), would have clues such as:

- (1) Is switch S1 open when load L1 is desired?
- (2) Is switch S1 closed when load L1 is not desired?
- (3) Does the label S1 reflect the true function of L1?
- (4) Et Cetera.

Obviously, sneak circuits are rarely encountered in this topograph because of its simplicity. Of course, this is an elementary example and is given primarily as the default case which covers circuitry not included by the other topographs.

With each successive topograph, the clue list becomes longer and more complicated. The clue list for the 'H' pattern includes over 60 clues. This pattern, because of its complexity is associated with more sneak circuits than any of the previous patterns. Almost half of the critical sneak circuits identified to date can be attributed to the 'H' patterns. Such a design configuration should be avoided whenever possible. The possibility of current reversal through the 'H' crossbar is the most commonly used clue associated with 'H' pattern sneak circuits and so will be illustrated in the example below.

#### THE RELUCTANT REDSTONE

The circuitry presented in Figure 2 represents part of the ignition control circuitry for a Redstone booster used during the Mercury Program in the early 1960's. This schematic would not be encoded for sneak circuit analysis because it is not considered to be a "manufacturing level" schematic. It does serve to illustrate, however, the difficulty involved in separating a given circuit from its surrounding circuitry and simplifying it to the extent necessary for an effective analysis. As can be seen in one of the network trees for this circuit shown in Figure 3, the computer removes all connectors, terminal blocks,

unnecessary nodes, splices, and any other extraneous circuit elements that do not affect the actual electrical continuities involved. Only active circuit elements remain in the completed network trees. When the network tree has been obtained, a sneak circuit analyst examines each node in sequence, determining which of the basic topographs the node is a part of and applying the clues that pertain to those topographs. After all such clues have been applied, all sneak circuits present in that network tree will have been identified. In the Redstone booster case, the pertinent clue is the 'H' pattern clue relating to reverse currents as previously discussed. This clue would be stated as:

"Is it possible for current to flow in both directions through the 'H' cross bar?"

If the answer to this question is yes, a sneak circuit is not automatically indicated as this may have been the design intent. The analyst must determine whether or not this is a desired condition within all foreseeable circumstances. In the case of the Redstone circuit, the condition definitely was not desired. The design intent was that the Cutoff Command relay contacts or the Pad Abort switch should energize the Engine Cutoff coil and the Abort Indicator coil. When the Launch Command contacts close, they should turn on the Launch indicator lights only. The possible reverse current can only exist when the ground below the indicator lights is lost. This would occur if the Tail Plug umbilical opened before the Control umbilical separated. If this happened, current would flow through the Launch Command relay contacts, the Launch indicator lights, through the suppression diode of the Abort Indicator coil and finally through the Engine Cutoff coil to ground. By this means, the Launch Command can activate the Engine Cutoff coil if the Tail Plug umbilical separates before the Control umbilical.

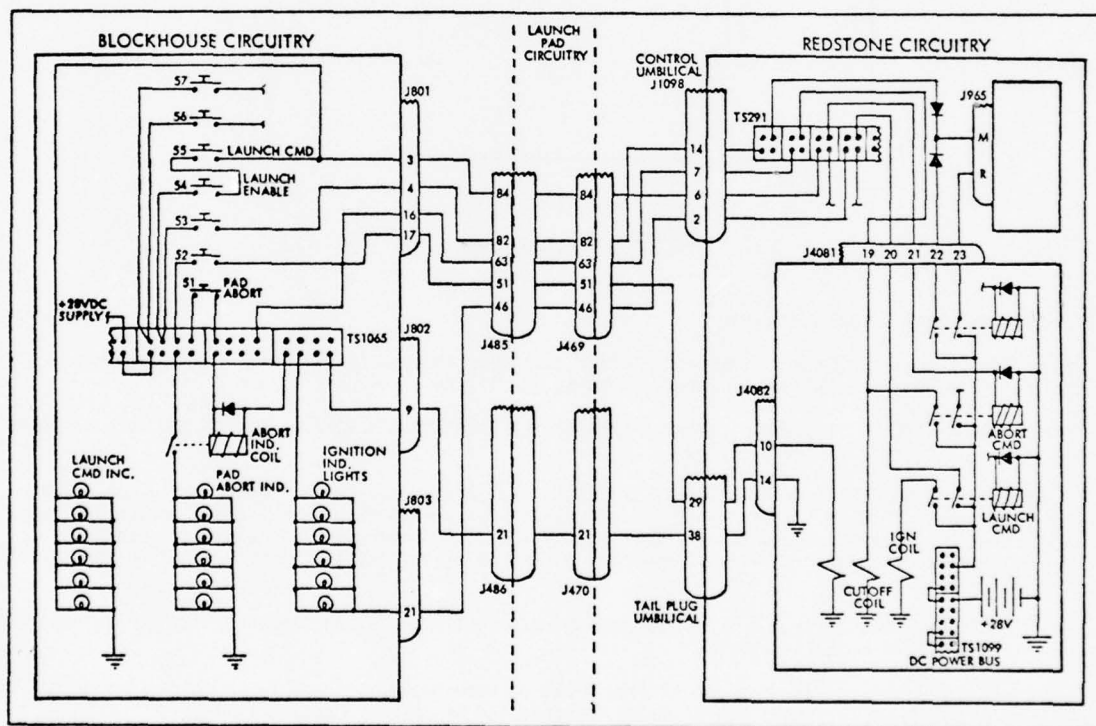


FIGURE 2

#### RELUCTANT REDSTONE SIMPLIFIED SCHEMATIC

Unlikely as this event may appear, it did occur on November 21, 1961. After more than 50 sequentially successful Redstone booster launches, a Redstone booster with a Mercury capsule on it was to be launched. When the ignition command was given, the booster fired. After lifting several inches from the pad, the engine inexplicably cut off. The booster settled back on the pad. The Mercury capsule jettisoned and deployed its parachutes. Damage was slight, but a highly explosive rocket with no means of control sat on the pad for 28 hours. No one dared to approach the Redstone booster until its batteries drained down and the liquid oxygen evaporated. Later investigation revealed that the Tail Plug umbilical had separated 29 milliseconds prior to Control umbilical disconnect. This was enough time for the sneak circuit shown in Figure 3 to occur and abort the mission.



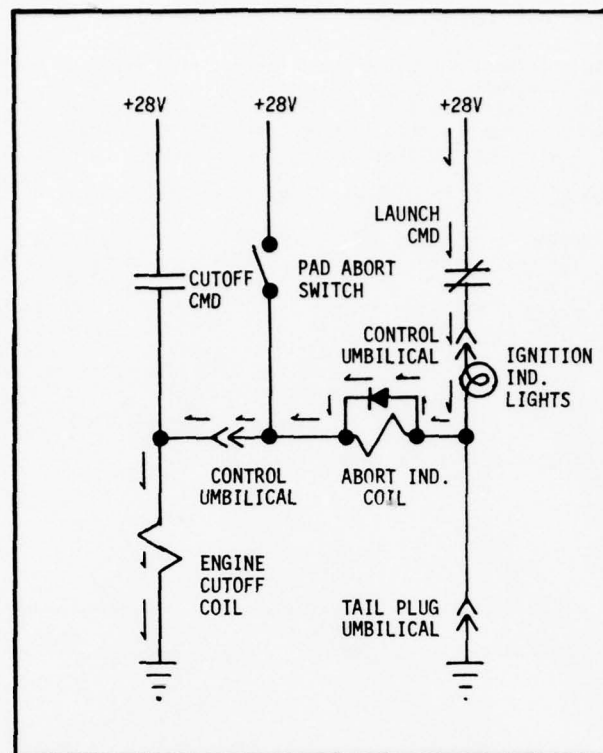


FIGURE 3  
REDSTONE NETWORK TREE

#### SNEAK CIRCUIT ANALYSIS REPORTS PRODUCED

Sneak circuit analysis of a system produces the following three general categories of outputs.

- (1) Drawing Error Reports, (2) Design Concern Reports and (3) Sneak Circuit Reports.

DRAWING ERROR REPORTS disclose document discrepancies identified primarily during the data encoding phase of the Sneak Circuit Analysis effort.

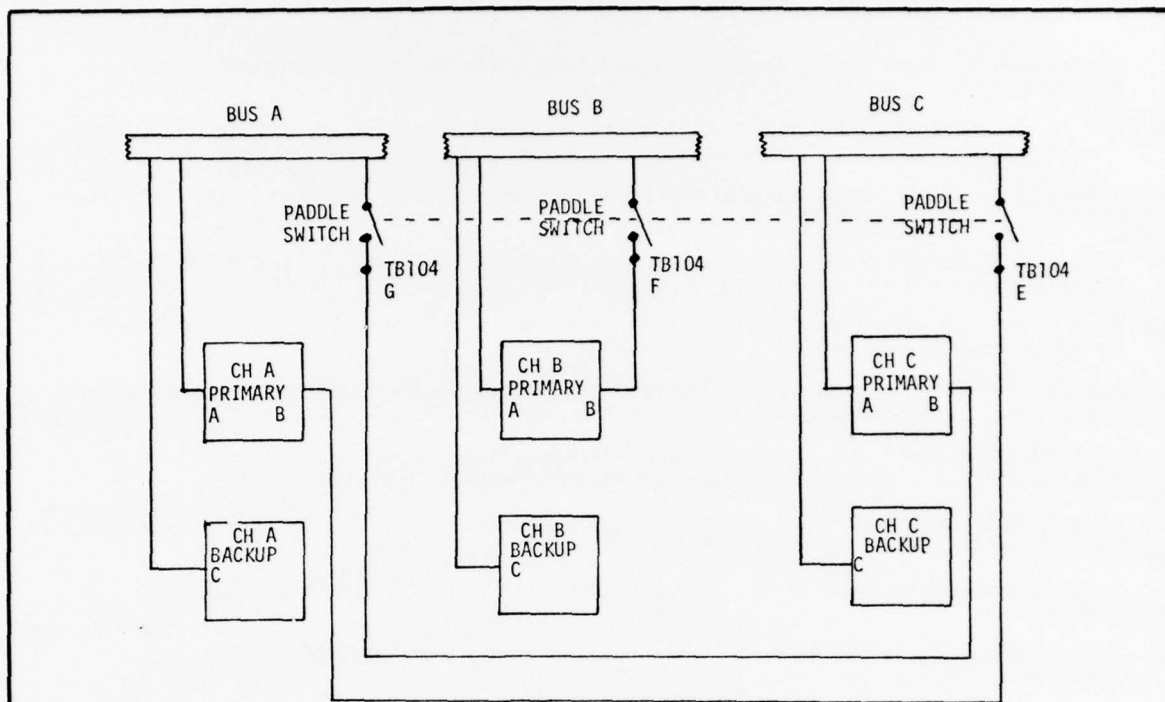
DESIGN CONCERN REPORTS describe circuit conditions that are unnecessary or undesirable but which are not actual sneak circuits. These would include single-failure points, unsuppressed inductive loads, unnecessary components, and inadequate redundancy provisions. A number of these kinds of conditions usually are identified whenever an analyst examines a circuit at the level of detail required for a formal Sneak Circuit Analysis.

SNEAK CIRCUIT REPORTS delineate the sneak conditions identified during the analysis. These reports typically fall into four broad categories:

- (1) Sneak Paths which allow current to flow along an unsuspected path or in an unintended direction.
- (2) Sneak Timing which causes functions to be inhibited or to occur unexpectedly.
- (3) Sneak Labels on switches or indicators which cause incorrect actions to be taken by operators.
- (4) Sneak Indications which cause ambiguous or false displays.

Sneak Circuit Analyses have been performed on more than fifty complex electrical systems. These analyses have included such diverse projects as Apollo, Space Shuttle and F-8 Digital Fly-By-Wire for the National Aeronautics and Space Administration; Pershing Missile and F-4C Flight Control System for the military; the N-Reactor emergency control and shutdown systems in the Nuclear Power field; and the Thistle Field A Platform upending and placement control systems for the oil industry.

Figure 4 illustrates a typical control system Sneak Circuit (identified during the analysis of a Digital Fly-By-Wire Flight Control System). Sneak Circuit Analysis is applicable to almost any complex electrical system involving either analog or digital circuitry or both. Software Sneak Analysis employs an analogous method developed for analyses of complex software systems. Without exception, critical sneak circuit conditions have been identified in every system investigated, even those that have been thoroughly analyzed by other means, comprehensively tested, simulated, breadboarded, and operated for a number of years. Sneak circuits are apparently universal, and Sneak Circuit Analysis can and does find them.



#### DESCRIPTION

The correct operation of each of the primary systems illustrated required power at both points A and B. The backup systems require power at the points labeled C. The Channel B power feed circuitry shown above is typical of how all three channels should have been wired. The wires from TB104 pins G and E of the paddle switches to Channel A circuitry and Channel C circuitry respectively were inadvertently reversed. This wiring error would result in the loss of both Channel A primary and backup systems and the Channel C primary system through a single bus failure involving Channel A. Due to the voting logic of the primary system, the Channel B primary system is automatically disabled when the other two primary systems are lost. A failure involving the Channel C bus would have a similar effect.

A single failure, therefore, involving Bus A will disable the entire primary system and the Channel A backup system. A single failure involving Bus C will disable the entire primary system and the Channel C backup system.

FIGURE 4

SNEAK PATH IN FLIGHT CONTROL SYSTEM PRIME POWER DISTRIBUTION

REFERENCES

- (1) Rankin, John P., Sneak Circuit Analysis, Nuclear Safety, Vol. 14, No. 5, pp. 461-467, (Sept-Oct, 1973)
- (2) Hill, E. J. and Bose, L. J., Sneak Circuit Analysis of Military Systems, Proceedings of the 2nd International System Safety Conference, pp. 351-372, (July, 1975).
- (3) Clardy, R. C., Sneak Circuit Analysis Development and Application, 1976 Region V IEEE Conference Digest, pp. 112-116 (April, 1976).
- (4) Sneak Circuit Analysis of the AEC DuPont Savannah River Plant Automatic Incident Action System, Boeing Aerospace Company, Houston, Texas, December 1972.
- (5) Sneak Circuit Analysis of Morgantown PRT Engineering Station and Vehicle Report D2-118467-1, Boeing Aerospace Company, Houston, Texas, July 1973.
- (6) Pershing Missile System Sneak Circuit Analysis Final Report, Report D256-10004-1, Boeing Aerospace Company, Army Systems Division, Huntsville, Alabama, December 13, 1973.
- (7) Sneak Circuit Analysis of N Reactor (Prepared for Atomic Energy Commission, Richland Operations Office), Report D2-118542-1, Boeing Aerospace Company, Houston, Texas, July 31, 1974.
- (8) Sneak Circuit Analysis of F-4C Flight Control System, Report D2-118545-1, Boeing Aerospace Company, Houston, Texas, September 1974.
- (9) Skylab Saturn Workshop Sneak Circuit Analysis Report D2-118461-1, Boeing Aerospace Company, Houston, Texas, May 1973.
- (10) Sneak Circuit Analysis of B-52/FB-111 AMAC and Release Circuitry, Report D2-118576-2, Boeing Aerospace Company, Houston, Texas, October 1975.
- (11) 747 Landing Control and Logic Unit Sneak Circuit Analysis, Report D2-118473-14, Houston, Texas, November 1973.
- (12) Sneak Circuit Analysis of Thistle Field "A" Platform, Boeing Aerospace Company, Houston, Texas, March 1976.
- (13) "First Mercury-Redstone Flight Test Fails On Pad", Aviation Week, November 28, 1960.
- (14) "Second MR-1 Test Planned in Two Weeks", Aviation Week, December 5, 1960.

# BUILT-IN TEST TECHNIQUES FOR DIGITAL FLIGHT CONTROL SYSTEMS

by  
W. A. Plice  
Senior Project Staff Engineer  
HONEYWELL INC.  
1625 Zarthan Avenue  
St. Louis Park, Minnesota 55416  
U. S. A.

## SUMMARY

The techniques discussed in this paper are suitable for use while the flight control system is performing its normal task. Most of these techniques are also applicable on the ground.

Since many inputs and outputs of a digital flight control system are analog signals, some analog testing capability is required. The basic concepts of analog testing may often be carried into digital testing.

Monitoring involves comparison of the performance of the item under test to the performance of a model. If the model is dynamic, it is a suitable standard of performance over a wide range of operating conditions; if the model is fixed, it is a suitable standard of performance only under a specific set of operating conditions.

Where redundant information is available in the system, the system itself may form all or part of the model. Computations can extract model information from signals that contain composite data. A single accelerometer displaced from the center of gravity can, for example, be used to verify the performance of a rate gyro and an accelerometer as well.

Setting tolerances on signal levels for good-bad decisions is a difficult problem often solved empirically. Parameter estimation techniques permit detection of component electrical parameter shifts. The tolerances used in good-bad decisions are then those of component values which are independent of signal amplitudes. The parameter estimation test technique also provides automatic fault isolation.

Stimulated monitoring is possible where the item under test is time multiplexed or where the stimulus can be designed to have negligible effect on the system performance.

Software simulation is a useful modeling technique for digital item testing. Where stimulated testing is possible, the model need be only a set of input/output patterns. Dynamic digital models which monitor digital items on line are complex and tend to require large amount of computer time.

Fixed models, such as parity checkers and watch dog timers, are frequently used in digital monitoring.

Wrap-around testing is a practical preflight and post-flight test technique.

## INTRODUCTION

Built-in test capability in a digital flight control system can reduce support costs, reduce turn-around time and improve mission success probability. Because of these potential benefits BIT (built-in-test) is increasingly incorporated as an integral feature of system design. Where redundancy exists, comparison testing of corresponding signals or parameters of the redundant elements is an effective built-in-test technique. Since comparison of redundant modules has had ample exposure in the fault-tolerant literature, this paper will emphasize other techniques for in-flight testing of digital flight control systems. The emphasis in this paper is upon techniques which are applicable in flight. These same techniques are usually applicable on the ground.

## DEFINITIONS

For the purposes of this paper, testing will be defined as a process designed solely for the purpose of determining whether an item is functioning or is capable of functioning within acceptable limits. The testing process may include one or more elemental tests which are defined as the process of determining whether a measured quantity lies within a range of values.

On-line testing is that testing which is carried out while the item is available for its normal function and does not delay or disrupt the execution of that normal function.

Monitoring is on-line testing which does not involve the use of overt test stimuli. Testing which involves the use of stimulus superimposed on the nominal signal, but not affecting the system performance, may also be included in the monitoring category.

Comparison testing is testing which involves direct comparison of a parameter or signal of an item to a standard like parameter or signal.

Reasonableness testing is testing which can detect malfunction of the equipment under test only when that malfunction causes the signal or parameter in question to fall outside of a range of values. The test is inconclusive for signals or parameters inside the "reasonable" range of values.



In-line testing is that testing which can be performed by a unit on itself without reference to a redundant item, built-in-test equipment, or external test equipment.

Functional testing is that testing which establishes whether an item under test functions within specified performance limits.

Fault isolation testing is that testing which establishes the physical location of the failure within a faulty item.

Dynamic model is defined as one whose transfer characteristics is a good approximation to that of the unit it models over a wide range of input conditions.

Fixed model is one which is a valid representation of the item under test only for a specific set of fixed input values.

## IN-FLIGHT MONITORING TECHNIQUES

Monitoring utilizes the information which can be obtained from the item while it is in its normal use. Monitoring does not introduce any significant perturbations into the behavior of the item in question. Monitoring may require the use of test equipment.

### Analog Monitoring

Since most sensors used by digital flight control systems are basically analog devices, some consideration of analog monitoring techniques is necessary. Where the analog device, circuitry or equipment is redundant, comparison of corresponding signals or parameters of the redundant units is valid and cost effective. Where such redundancy does not exist, one or more of the following techniques may be useful:

- Dynamic Model - Input and Output Accessible -- When both input and output of an analog device are available, it is possible to model the unknown device and perform a reasonableness test by comparing the results obtained from the model with the actual signal. The model may be a hardware or a software model. When the model is not continuously connected as might be the case with a software model or a time-shared hardware model, care must be taken to account for the state of energy storage elements within the flight hardware at connection time. Figure 1 shows the dynamic model concept applied to two simple analog signal processing elements. In Figure 1a an inverse model is used since that model is differentiating and avoids the initial condition problem of an integrating model.

In Figure 1b a forward model is used because the flight hardware is basically a high pass. Here the initial condition problem is not severe because the time constant  $T_3$  is relatively short.

In both cases of Figure 1, limits are established in terms of a percentage of value rather than a fixed magnitude tolerance band. If the signal falls outside of the "GO BAND" the output logic level will go to a logic zero. An integrating block is shown on the output logic to eliminate any noise pulses which may otherwise produce false failure indications.

The BIT models of Figure 1 may be implemented either in hardware or software. Where general-purpose computing capability is available during the BIT time, software models may be used. Hardware models may utilize either analog circuitry or hardware digital filters. The absolute value computations and the comparisons are conveniently done in the computer rather than in hardware.

- Dynamic Model - Input Inaccessible -- When the input is inaccessible as, for example, in inertial sensors, an alternate source of information is needed to permit dynamic monitoring. This information must ultimately come from an independent sensor. In some cases, a single independent BIT sensor can be used to test more than one flight sensor. This is the case in Figure 2 where an inner stabilization loop uses a rate gyro and an accelerometer. Since an accelerometer mounted in a position which is displaced from the center of gravity along an axis in the plane of rotation will sense acceleration which has both rotational and translational components, it will provide information adequate to verify both the rotational rate gyro and accelerometer which are sensitive to the same inputs.

In Figure 2 the primary flight sensors are used to compute the quantity

$$q \left( K_1 + \frac{C_1 K_2 S}{TS+1} \right) + a \left( \frac{K_2}{TS+1} \right)$$

where  $q$  is the rotational rate and  $a$  is the linear acceleration. Two other outputs are computed, using an additional accelerometer, which are essentially duplicates of the primary output. Since each of the three outputs depends upon only two sensors, if one output does not agree with the other two a fault is detected and may in fact be isolated to a specific sensor as shown in Table 1.

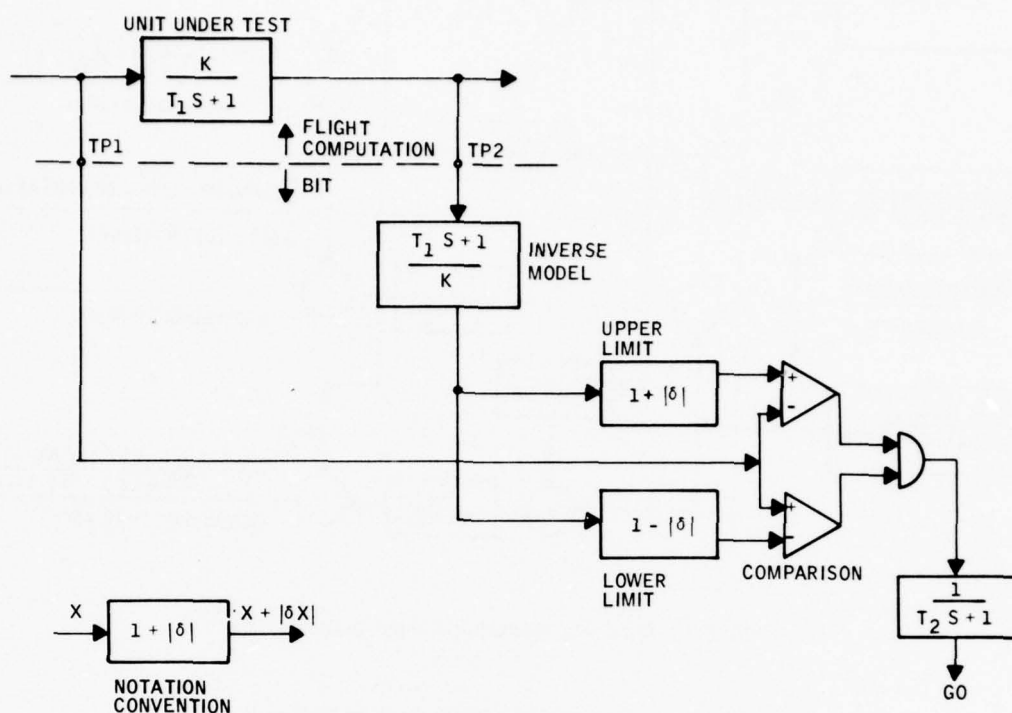


Figure 1A. Inverse Model Input and Output Accessible

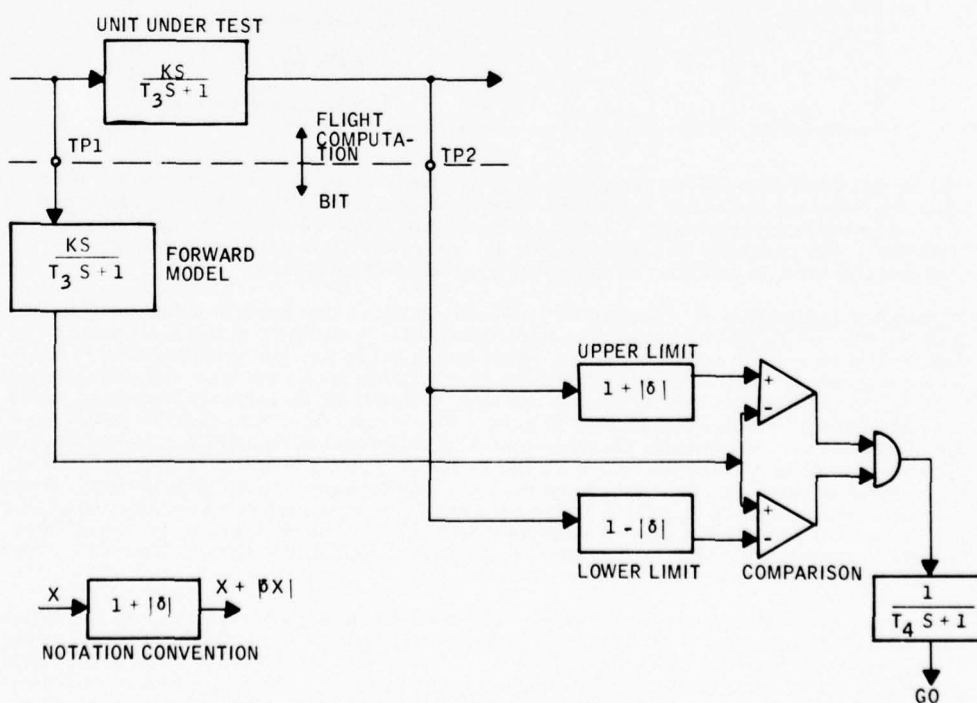


Figure 1B. Forward Model Input and Output Accessible

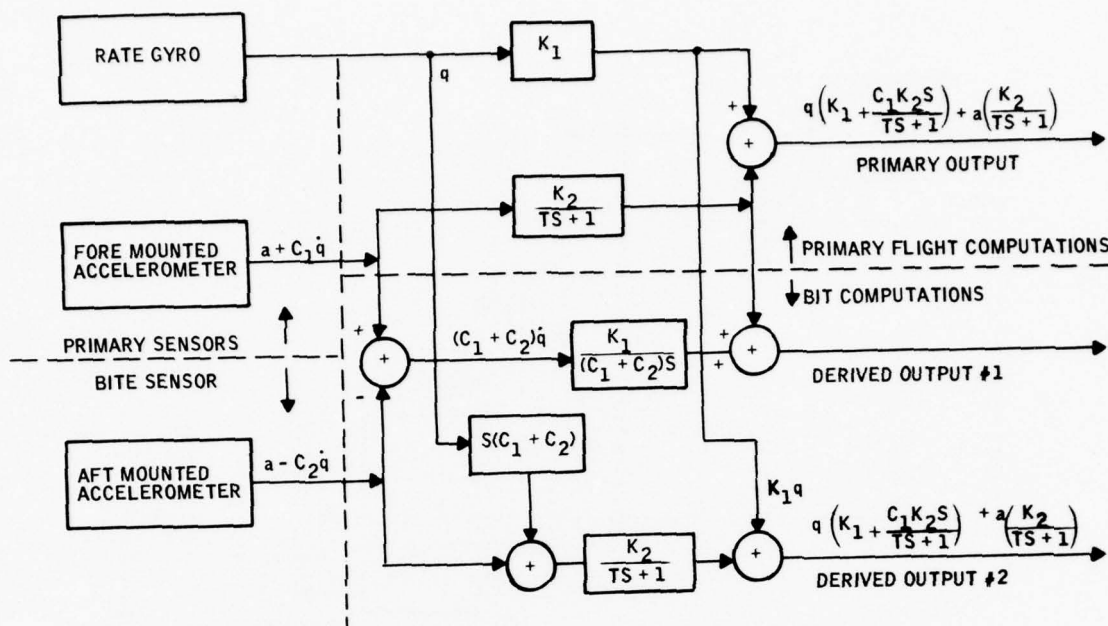


Figure 2. Dynamic Modeling - Input Inaccessible

Table 1. Fault Isolation Properties of Figure 2

Output States: 0 = Minority  
1 = Majority

Primary	Derived #1	Derived #2	Fault
1	1	0	Fore Accelerometer
1	0	1	Rate Gyro
0	1	1	Aft Accelerometer

The comparison of the three outputs may be carried out as shown in Figure 3. Here again the tolerance band for reasonableness testing is established as a percentage of one of the signals being compared. Logic circuitry identifies the input which is in the minority. The comparison and logic may be implemented in hardware, but they are conveniently done in software if computer capability is available.

- Parameter Estimation** -- The dynamic model techniques use models which have fixed parameters, or at most programmable parameters. With these models, a fault is detected when signals which should be equal are found to deviate by more than an established tolerance level. The establishment of a suitable tolerance that will avoid nuisance trips but yet detect failures early is a difficult problem which can only be solved in the context of the particular application at hand. There are, however, definite tolerances on the electrical and mechanical characteristics of individual components of a system. Since the parameters of a transfer relationship are known functions of the component values, it is possible to establish specific limits for the transfer relationship parameters. When the transfer relationship is specified in factored form, each parameter is a function of only a few components in the system. These functions are context free and limits on system transfer parameters which represent the extremes obtainable with in-tolerance components can be readily estimated.

Parameter estimation techniques can be used to compute the present value of the transfer parameters of an analog system using the normal signals of the system. One technique of parameter estimation is shown in Figure 4. Here the parameter of a software model of the unit under test is adjusted by a parameter adjustment algorithm which minimizes the mean squared error of the output of the model as compared to the measured output of the unit under test. The use of this technique requires considerable computation capability, but the computing time may be acceptable when operated in a tracking mode; that is, using the last estimate of the values of the parameters as the starting point for the next estimation in the next BIT interval.

Parameter estimation can be used for fault isolation using the parameter values as a fault signature. Figure 5 shows a simple linear circuit which may be fault isolated by use of Table 2. In the construction of Table 2, if a parameter is more than 20 percent higher

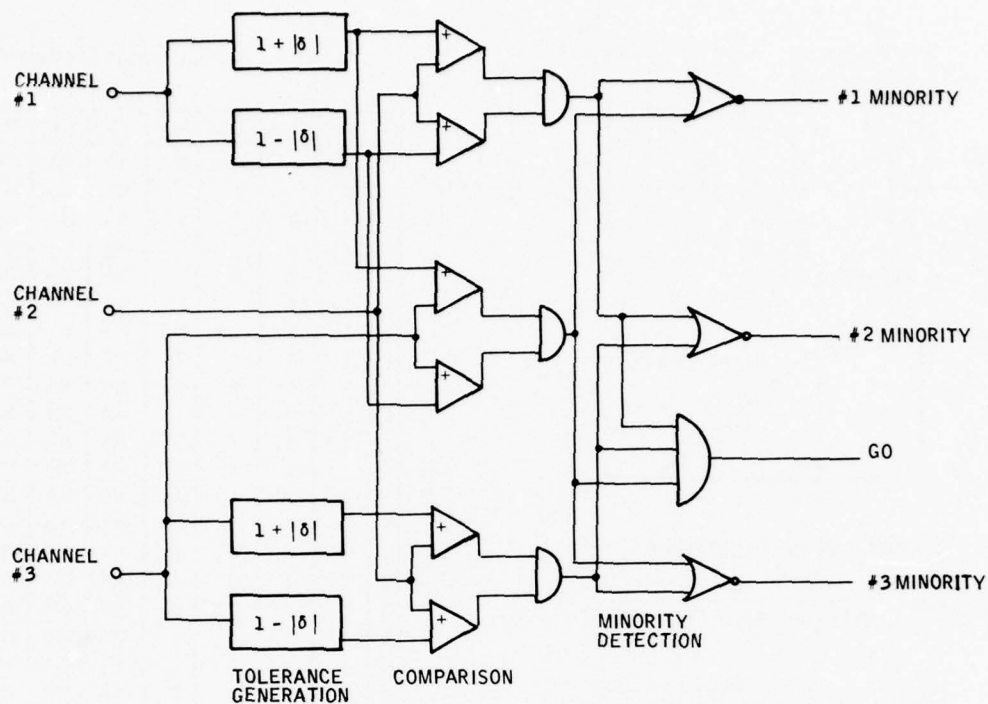


Figure 3. Channel Comparison Logic

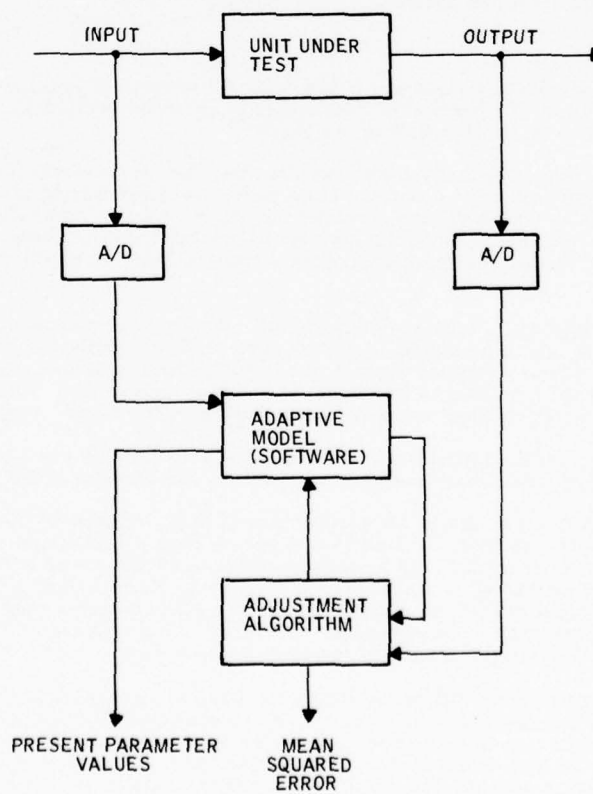
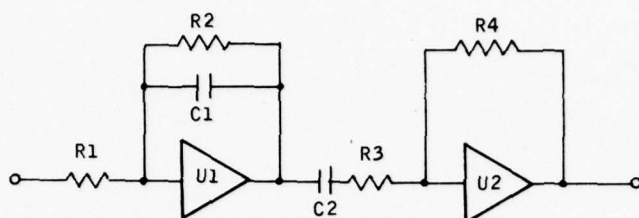
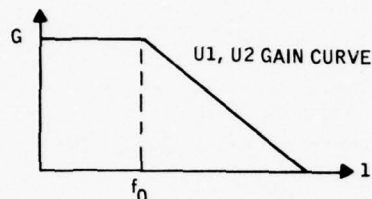


Figure 4. Parameter Estimation





a) CIRCUIT DIAGRAM



b) AMPLIFIER CHARACTERISTIC

$$F(p) = \frac{Kp}{(p + A_1)(p + A_2)(p + A_3)(p + A_4)}$$

OR:

$$F(p) = \frac{Kp}{(p + A_1)(p + A_2)(p^2 + pB_3 + B_4)}$$

c) TRANSFER FUNCTION

Table 2. Analog Fault Dictionary

K	A1	A2	A3	A4	B3	B4	Replace
2	2	2	1	1	NA	NA	C1
2	2	1	1	1	NA	NA	U1
2	2	0	1	1	NA	NA	R1
2	1	2	1	1	NA	NA	C1
2	1	1	2	1	NA	NA	U2
2	1	1	1	1	NA	NA	R1
2	1	1	NA	NA	2	2	R3
2	1	1	0	2	NA	NA	R3
2	1	1	0	1	NA	NA	R4
1	2	2	1	1	NA	NA	R2
1	1	2	1	1	NA	NA	R2
1	1	1	1	2	NA	NA	C2
1	1	1	1	1	NA	NA	No Fault
1	1	1	1	0	NA	NA	C2
1	1	0	1	1	NA	NA	R2
0	1	1	2	1	NA	NA	R4
0	1	1	2	0	NA	NA	R3
0	1	1	1	1	NA	NA	R1
0	1	1	0	1	NA	NA	U2
0	1	0	1	1	NA	NA	C1
0	0	1	1	1	NA	NA	U1
0	0	0	1	1	NA	NA	U1

Figure 5. Typical Linear Circuit

than nominal it is given the weight 2. If less than 80 percent of nominal value it is given the weight 0. If within  $\pm 20$  percent of nominal it is given the weight 1. In this case the faulty component can be located with no ambiguity.

- **Stimulated Monitoring** -- In many flight systems the signals present in an item may be essentially quiescent for long periods of time with only short bursts of significant activity. Monitoring techniques which depend upon the use of the signals normally existing in the item under test may produce inconclusive results during these periods of inactivity. Since an inactive period may be the most appropriate time to test, stimulated monitoring may be considered.

In stimulated monitoring, a small energy signal, usually of zero mean, is superimposed upon the normal signal. The signal is designed to produce negligible effect upon the performance of the system. Typical signals are high-frequency or d-c tracer signals, where appropriate, or doublet pulses to introduce cancelling transients. When manual involvement is possible, an intentional maneuver can introduce the signal activity need.

All of the dynamic model testing techniques are suitable for use when a stimulus signal is present. The use of a stimulus makes it possible to use simpler fixed models as well.

- **Fixed Models** -- A fixed model is simply a nominal fixed value with fixed limits against which the signal obtained from the unit is compared when all assumed operating conditions are fulfilled. The fixed model may be implemented in hardware or software. The use of a fixed model for monitoring is illustrated in Figure 6. In this case a low-energy signal is within the pass band of the unit under test but is rejected by the rest of the system. Again the comparison may be hardware or software. This test may be sensitive only to a subset of the possible failure modes of the item being tested.

Sometimes the monitor may utilize an attribute of the signal which is not used to convey information in the normal use of the signal. An example of this is shown in Figure 7. The purpose of the monitoring in this case is to detect intermittants. The signal is a modulated carrier. Transients are detected by a pair of counters. One counter counts the transitions of the basic frequency source, the other counts the transitions of the monitored signal. If the counters do not agree over a given time period, a failure has occurred.

#### Digital Monitoring

Special-purpose digital hardware can frequently be described by a reasonably tractable fixed input-output relationship. Where this is the case, most of the monitoring techniques discussed under analog monitoring have their counterparts for monitoring digital items. This is true, for example, where hardware digital filters are used to implement signal processing functions. The same reasoning applies to software modules which have fixed input-output relationships such as an integration routine. Software

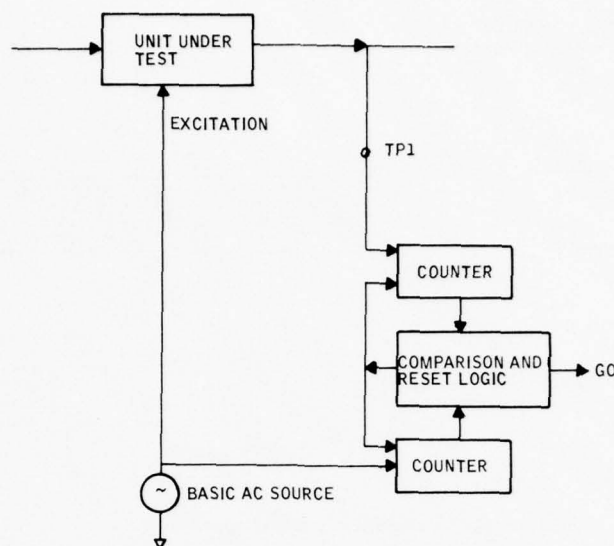


Figure 6. Fixed Limit Stimulated Monitoring

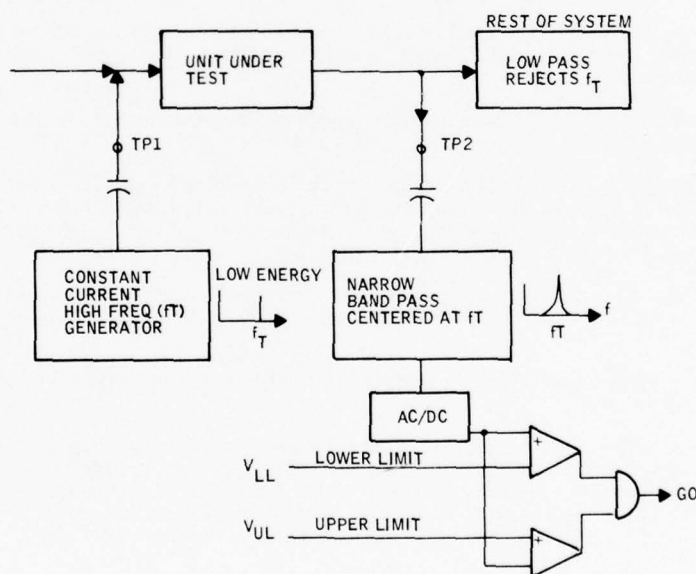


Figure 7. Intermittent Detection Fixed Model Monitoring

of this type can be monitored by appropriate modifications of the techniques discussed under analog monitoring. The discussions of this subsection will be, therefore, restricted to those techniques which are peculiarly suitable to testing digital items.

**Dynamic Modeling.** Digital hardware can, of course, be modeled by duplication. This provides hardware redundancy and can be used to improve the system reliability. Lower-cost alternatives are time-shared reconfigurable hardware and software modeling. Since time-shared models and software models (that is, part-time models) do not usually experience the full history of the item under test they must have some provision for determining the internal states of the item under test. In the case of a general-purpose computer the number of internal states is extremely large.

To avoid undue complexity and difficult state initialization, digital systems are usually modeled as a collection of smaller models. The test comparison spans only the hardware or software represented by a single model. Dynamic models are used only where the number of states is small and state information is readily available. This piecewise modeling has the advantage of inherent fault localization. Where the number of internal states increases, dynamic modeling becomes less useful.

A possible use of software dynamic modeling is shown in Figure 8. Here a test processor contains a software simulation of the item being tested. The model must permit unknown states (ternary simulator).

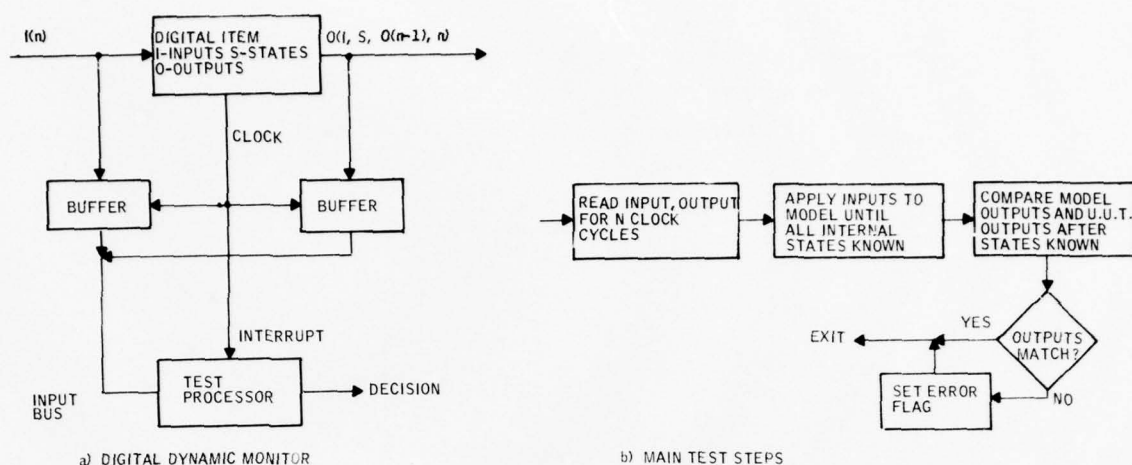


Figure 8. A Digital Dynamic Monitor

The test processor first acquires a number of samples of the inputs and outputs of the item. The inputs are then applied sequentially to the software model until all of the states have been established. Obviously, any information about internal states gleaned from the outputs of the item is also used.

After the internal states have been established, the remaining inputs are applied to the model and the simulated outputs compared to the corresponding output acquired from the item under test. If a mismatch occurs an error has been detected.

The test processor of Figure 8 can be slow since it operates on stored data, and it may be time shared among several test problems.

**Echo Tests.** Good digital system and software design practice which has evolved to provide error-free communication between the central processor and asynchronous peripherals inherently includes test provisions. These same provisions with some expansion can be used for BIT. One general class of tests used to verify correct communication may be termed echo tests.

A typical echo test is that used in magnetic tape units. In this case a read head reads from the tape the data which was just written by the write head. Simple comparison of the read and write data verifies correct operation.

Another echo test is the requirement that an addressed peripheral respond with a "ready" signal before data is sent by the processor.

Expansion of the echo test to more completely verify a digital function may be performed by the hardware shown in Figure 9. While this requires some additional hardware, the hardware is simple and inexpensive and its presence permits complete checkout of the selection of one of  $n$  analog data sources for the A/D converter. The use of a separate data buffer and address decoder permits verification that only one of the  $2^n$  switches is commanded "on" and that the rest are commanded "off."

**Stimulated Monitoring.** Many digital applications involve time multiplexing of some part of the system such as a data bus or processing unit. In these cases a time slot is usually available for test activity. Other elements of the system have inactive periods during which testing can be performed.

In either case it is usually permissible to inject stimuli or otherwise exercise the item during the test period. The test procedure must be such that no information is lost as a result of changing the state of any memory elements in the item under test.

A stimulated monitoring technique is illustrated in Figure 10. The test processor uses a set of "test vectors" generated off-line and verified by off-line simulation to detect a high percentage of the possible faults of the unit under test. The test processor applies the input vectors in the prescribed order and compares the actual response vectors to the stored "good" responses. The test vectors provide an initializing sequence that bring the internal states to a known condition. Tests of this type may be performed by special-purpose hardware.

The test vector technique can also provide fault isolation to a small ambiguity region. The set of test vectors can be divided into groups for interleaving with the normal activity of the unit under test, provided an initializing sequence is provided for each restart point.

**Fixed Models.** The use of fixed models is one of the most prevalent techniques used in digital monitoring. Parity checking may be considered as fixed monitoring. In the parity case a signal attribute not used to perform any flight control function is utilized to monitor the performance of the item under test.

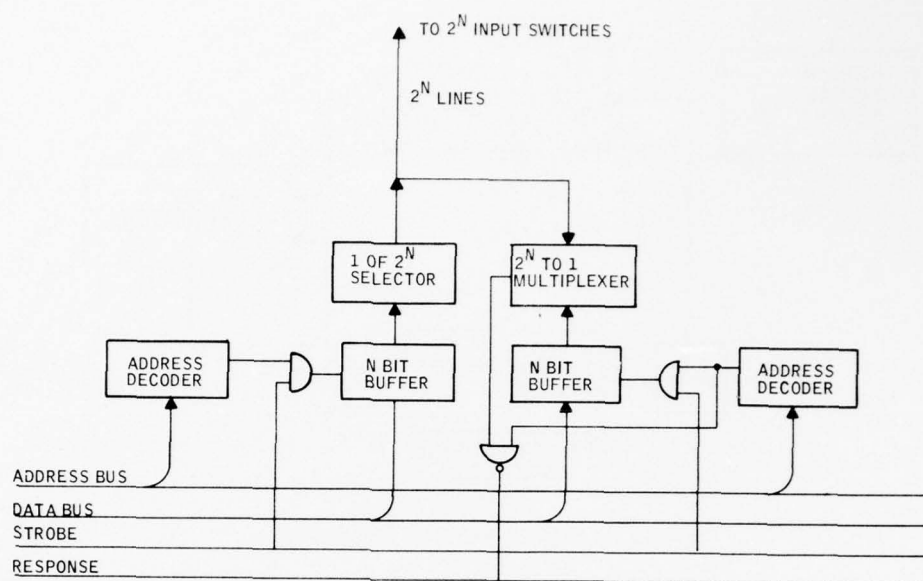


Figure 9. Typical Echo Check for BIT

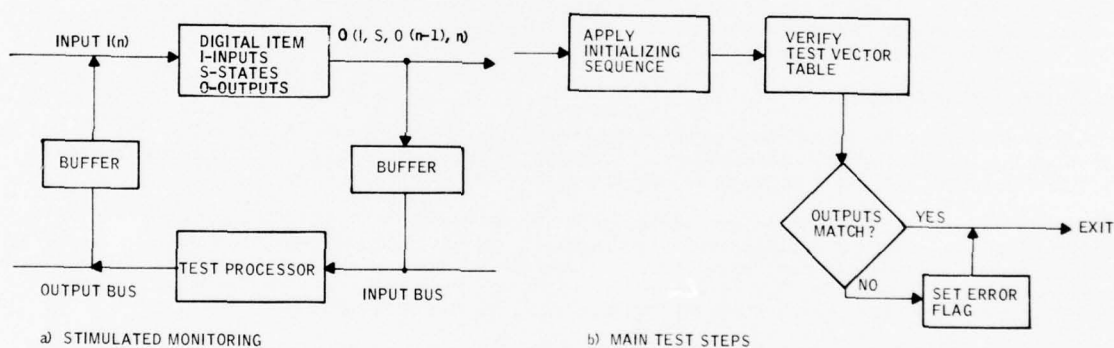


Figure 10. A Digital Stimulated Monitor

The watch dog timer is another example of fixed modeling. Properly functioning software in a properly functioning processor will reset the timer at fixed time intervals within a specified tolerance. An early or late reset will be interpreted as evidence of a malfunction.

Memory sum check involves a fixed model which is simply the sum of all the words in a section of memory when interpreted as binary numbers.

Sample problems interleaved with the operational program are a form of fixed model monitoring. Figure 11 shows a typical sample problem monitor used for digital flight control monitoring. The analog signal generator can be a very simple device since the only requirement is that the fundamental frequency be on one of the slopes of the notch. The analog filter must be stable in response shape and frequency. This monitor will check most of the instructions of the flight control processor as well as the arithmetic section, memory, I/O and the A/D and D/A converters.

**Self-Checking Circuits.** A self-checking circuit utilizes an error-detecting code. In fault-free operation, the output is always an acceptable code word. A detectable error will produce a non-code word. Self-checking circuits include checkers which will indicate the presence of non-code words. While self-checking will inevitably increase circuit complexity it need not result in a proportionate increase in failure rate, size or cost.

Implementation of self-checking at the integrated circuit level is very appealing. Many ICs are pin limited rather than complexity limited. It may be possible to obtain as many or nearly as many self-checking logic functions in an IC package as are now provided in non-self-checking logic. Since the failure rate (and cost) of ICs seems to depend more upon the number of pins than upon the internal complexity, the penalty paid for self-checking may be quite acceptable. Self-checking at the IC level has the added advantage of immediate fault isolation to the IC level.



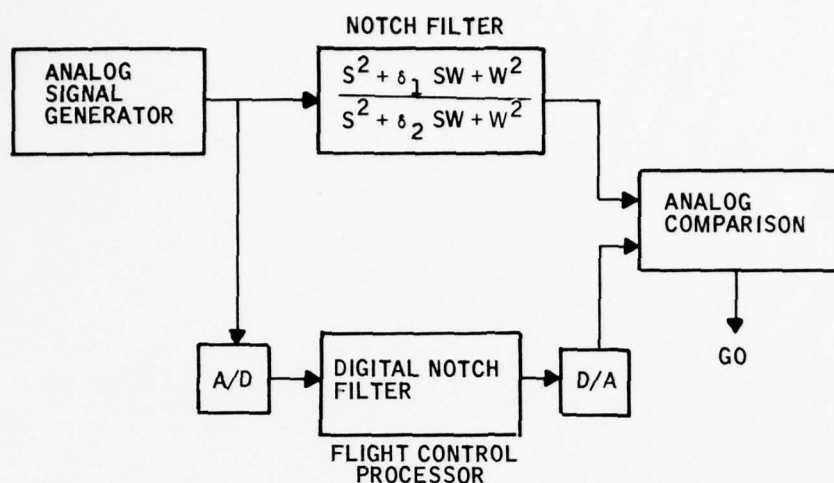


Figure 11. Sample Problem Monitoring

Unfortunately the state-of-the-art of self-checking circuits has not progressed to the point of widespread applicability. Simple fault detecting codes exist which are code preserving under arithmetic operations (exclusive OR), but no simple fault-detecting codes are code preserving under the logical operations (AND/OR). The work to date has generally used redundancy and comparison for testing these logic functions. Where the logic structure can be arranged to do so economically, arithmetic logic units which have an arithmetic mode can be used to implement AND/OR logic. These units are then placed in the arithmetic mode for self-checking.

#### Digital Memory Monitoring

Memory monitoring is treated as a separate topic because of the complexity and criticality of memory systems. Read/write memories are a particular problem since there is usually no absolute way to determine from alternate sources the correct value of stored data.

Complete memory redundancy at two, three, or higher levels is used for fault tolerance in critical applications. Where complete redundancy is not practical, partial redundancy is often used in the form of error detecting or detecting/correcting codes. Where the memory contents are fixed the memory sum check (a minimal redundancy) is often used. In semiconductor memories the effectiveness of error detection is strongly affected by the physical partitioning of the memory system.

**Single Bit per Word Parity.** A single bit appended to the word is set to zero or one as required to make the number of ones even (odd). An error in a single bit position will cause the parity to be incorrect. Single failures which cause simultaneous errors in more than one bit position may not be detected. For semiconductor memories, organizations in which each bit is produced by a separate chip are most suited to single-bit-per-word parity.

Where all bits of the word, including the parity bit, are produced by a single semiconductor chip, a single bit failure in address decoding will cause half of the words read to be incorrect. The parity bit in this case will not detect any of the errors. If the parity bit has a separate addressing structure, 50 percent of the data errors caused by a single address bit failure will be detected.

**Byte Wide Parity.** A common parity scheme provides one parity bit for each byte in the word. This scheme will detect all single bit errors per byte. If all bits of a byte, including parity bit, are on a single chip, no addressing failures will be detected. If the parity bits have a separate addressing structure, 50 percent of the byte addressing failures will be detected.

**Chip Wide Parity.** In this scheme, intended primarily for semiconductor memories, parity is computed over bit groups formed by taking one bit from the corresponding bit position in each chip forming the memory word. There are as many groups as bits per chip. Figure 12 illustrates chip-wide parity for a 16-bit word formed from memory chips four bits wide.

In the illustration, failure of the address decoding on a single chip will cause one group of four bits to be incorrect. Since this failure can cause at most one bit of a parity grouping to change, any data error due to address decoding failure will be detected. When the chips are one bit wide the chip-wide parity scheme reduces to a single parity bit per word, but any failure of a single chip that causes a data error will be detected.

**Error Correcting Codes.** Error correcting codes require greater redundancy than error-detecting codes and are inefficient in terms of memory size. For a 16-bit word, five Hamming parity bits are required for double-error detection or single-error correction and six bits are required if double-error detection and single-error correction is desired.

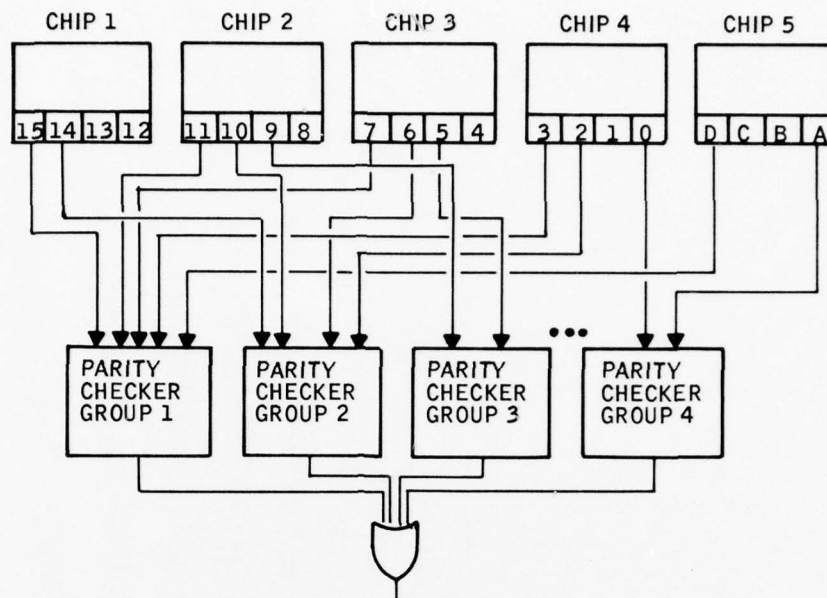


Figure 12. Example of Chip-Wide Parity Checker Architecture

**Memory Sum Check.** A common memory test technique is to form the correct sum of all the numbers that should be stored in a block of memory. This check sum is also stored (preferably in another block) for later use. When blocks are read out of memory or during a self-test cycle the denomination of all words in the block is computed and compared against the check sum word.

This technique is most applicable to read-only memories and mass storage memories. It can be applied, at considerable overhead to directly executing program memories.

Check sum provides poor coverage for the most significant bit position unless double-precision addition or end-around carry is used.

Where the carry digit is retained, sum check will detect 100 percent of all single-column data errors. For address errors (single address bit failed) the probability of data error detection is less than 100 percent. Where the address error affects only a single column, the probability of data error detection is only 80 percent for a 16-word group, 90 percent for a 64-word group.

#### PREFLIGHT AND POST-FLIGHT TEST TECHNIQUES

Most of the test techniques discussed as in-flight monitoring techniques are also applicable to post-flight and preflight testing. There is, however, in post-flight and preflight testing greater freedom to stimulate the item under test. Greater periods of uninterrupted time are available for post-flight and preflight testing. The use of computing capability of other on-board systems may be feasible ("global BITE").

Preflight and post-flight test objectives place greater emphasis on precision and fault isolation than is necessary for in-flight monitoring.

#### Wrap-Around Testing

A digital flight control system typically provides for analog, digital and discrete signal inputs and for similar outputs. For post-flight and preflight testing it is feasible to connect outputs to inputs and perform a complete end-to-end test of the complete digital flight control system with the exception of parts of some sensors.

A complete end-to-end test of a digital flight control system would begin with a self-test program which verifies the operation of the instruction set of the central processor. Testing would progress outward from this point, gradually adding elements of the system until finally transmission of information around the entire loop from processor to output to input to processor was verified. This process is illustrated in Figure 13. A carry-on operator control panel may be utilized if one is not an inherent part of the flight equipment. Embedded in the gross flow chart of Figure 13 are most of the test techniques discussed as suitable for in-flight monitoring.

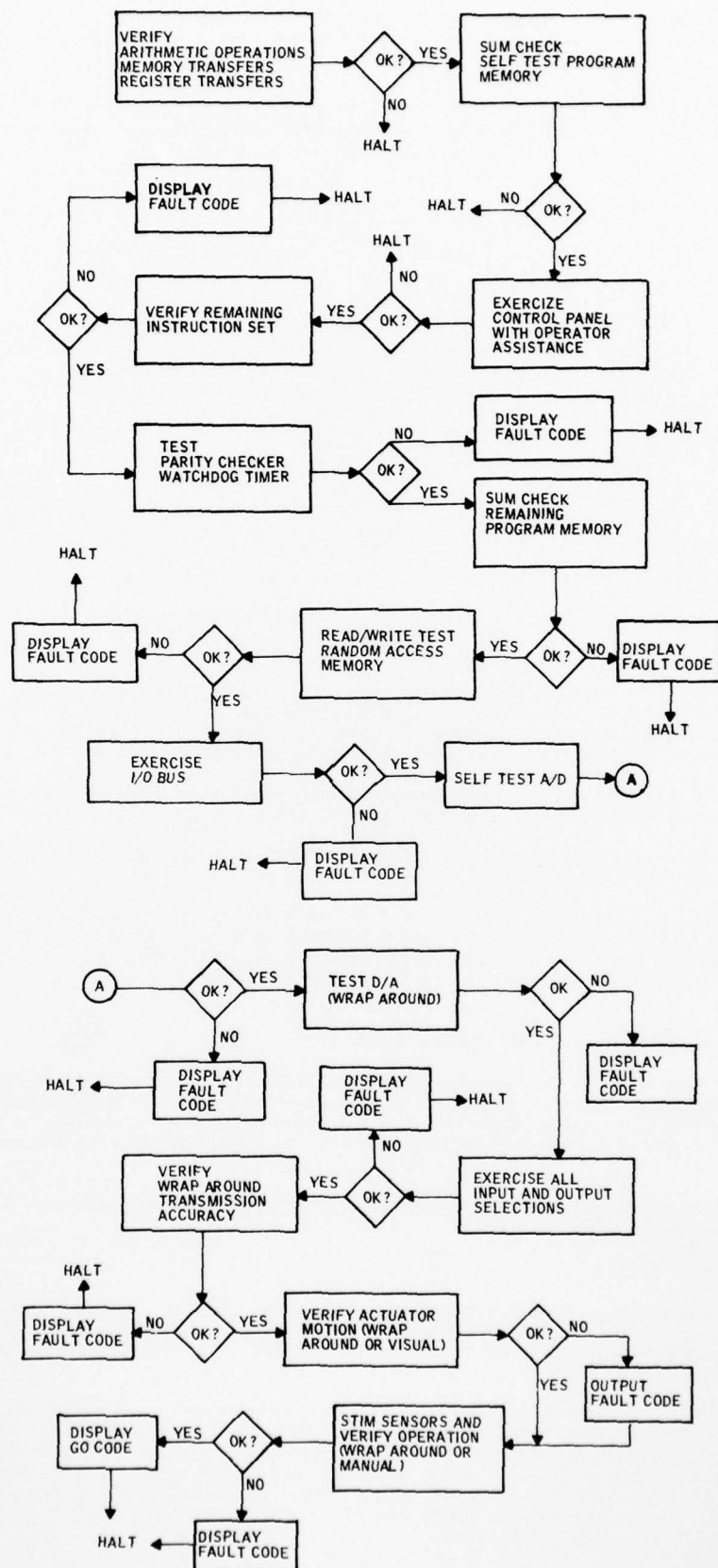


Figure 13. Wraparound Test

## CONCLUSION

The techniques useful for in-flight monitoring of digital flight control systems can be categorized as dynamic model or fixed model techniques. Built-in-test equipment utilizing these techniques can provide for in-flight and on-board testing with minimal carry-on equipment and no roll-up flight line equipment.

## REFERENCES

1. Honeywell Inc., "Advanced Fighter Digital Flight Control System (DFCS) Definition Study - Final Report," prepared for McDonnell Douglas Corp., St. Louis, Missouri, March 1975, Doc. W0728FR.
2. Seacord, C.L., and Vaughn, D.K., "Preliminary System Design Study for a Digital Fly-By-Wire Flight Control System for an F-8C Aircraft," NASA, Washington, D.C., January 1976, NASA Contractor Report CR-2609.
3. Jack, L.A., and Berg, R.O., "Coverage Analysis of Self-Test Techniques for Semiconductor Memories," Honeywell Inc., Systems and Research Center, Minneapolis, Minnesota 55413, Doc. MR12399, December 1975.
4. Benowitz, N., et al., "Fault Detection/Isolation Results From AAFIS Hardware Built-In Test," IEEE 1976 National Aerospace and Electronics Conference, Dayton, Ohio, May 18-20, 1976.
5. Plice, W.A., "Increasing System Reliability With BITE," IEEE NAECON, Dayton, Ohio, May 18-20, 1976.
6. Williams, M. and Angell, J., "Enhancing Testability of Large-Scale Integrated Circuits Via Test Points and Additional Logic," IEEE Trans. on Computers, Vol. C-22, January 1973.
7. Wakerly, J.F., "Partially Self-Checking Circuits and Their Use in Performing Logical Operations," IEEE Trans. on Computers, Vol. C-23, July 1974.



# 'Pre-Flight Dynamic Checkout'

by

D.R. Towill  
(U.W.I.S.T., Cardiff, U.K.)

Summary The advent of Fast Fourier Transform algorithms coupled to the increasing availability of data processing facilities has resulted in 'transfer function testing' increasing enormously in popularity. System impulse response, step response, or frequency response are all satisfactory dynamic signatures related to the transfer function of the system under test. As discussed in the paper, it is sufficient for many pre-flight checkout applications to estimate the signature at just a few carefully selected data points, thus considerably reducing test time and computational capacity required. This in turn permits the use of special purpose hardware such as Fourier Response Analysers when justified on cost-benefit or logistic grounds as an alternative to spectral analysis methods. The paper provides the basic ground rules for selecting system test stimuli and test features for both manual and automatic testing.

## 1. Introduction

Very many aircraft and avionic systems are required to suitably respond to some time varying input ('the message'), and to suitably reject other time varying inputs ('disturbances') in order to properly perform the operational role for which they are intended. For example in an automatic terrain following control system (ATFCS), the terrain profile plus set clearance represents the 'message', and the 'disturbances' will arise from such sources as sensor noise, internally generated noise due to discrete signals in the command computer, and wind gusts tending to deflect the aircraft off course. The total ATFCS is designed to adequately discriminate between the 'message' and 'disturbances' so as to achieve the desired level of mission effectiveness which in this case is the best balance between the probabilities of 'terrain clobber' and 'mission abort'.

In order to achieve an adequate design for the ATFCS, the system designer must represent both message and disturbances by realistic time varying signals. It is intuitively obvious (and substantiated by many company confidential system simulation studies) that dynamic tests are needed to rapidly establish the operational status of the system, and it is the purpose of this paper to review currently used techniques suitable for pre-flight dynamic testing. Although there is a swing towards computer controlled automatic test equipment (ATE) implementation of dynamic testing because of the advantage of speed at which test data becomes available, there are plenty of situations in which it is acceptable to have dynamic testing undertaken either in the manual or the built-in-test mode of operation. With one exception, all examples given in this paper are for 'open-loop' testing in that the aircraft is not part of the test loop. The system under test (SUT) may, of course, be a complex multi-loop feedback system which would naturally be tested with its own feedback paths closed. In breaking down such a complex system as the ATFCS for pre-flight testing, the dynamic test requirements must similarly be partitioned between the various SUT's such as compensation units, amplifiers, mixers, and servactuators in accordance with the permissible transient errors for each unit. All SUT's used as practical examples in the text represent analogue hardware, but the advent of digital autopilots should not invalidate the principles of dynamic testing for pre-flight checkout since preliminary studies have already indicated that hybrid systems may be tested using pseudo-noise signals and cross-correlation.(1)

## 2. What is Dynamic Testing?

The dynamic response of the system under test is defined as the behaviour of the system when stimulated by a time varying input such as the unit step or one of the many alternative signals which will be considered in a later section. Consequently, a dynamic test is any test which yields information on the dynamic response of the SUT, even if the data yielded does not completely describe the dynamic behaviour of the system. Thus if only the final value of the response of the SUT is measured, then the test would be classified as a static test only, whereas if the behaviour of the SUT is sampled at various times during the transient response, then the test would be classified as a dynamic test.

Fig.1 shows the step response of an autostabiliser unit, such as might form part of the ATFCS as recorded on an oscilloscope. Superimposed is a checkout 'mask' which the test technician uses to categorise the system into 'healthy' or 'sick' status according to whether or not the response crosses the boundary. In automatic tests, the response of the autostabiliser is sampled at a few discrete points in time,(2) and a judgement made by comparison with a set of checkout gates, or by reference to a decision surface, using, for example, the nearest neighbour rule. The choice of test features (in this example the features are the delay times at which the step response is sampled) is crucial in arriving at a high level of correct classification. In the language of pattern recognition theory, we are seeking test features which readily discriminate between 'sick' and 'healthy' systems, as shown in the two dimensional case of Fig.2(a), and wish to discard test features such as those shown in Fig.2(b) which confuse the status of the SUT. Fortunately, in avionics, unlike medicine, the system is created by a known designer, and mathematical and functional models exist. Providing the analysis is done at the system design stage, it is relatively straightforward to select adequate test features by simulating the response of samples of 'sick' and 'healthy' systems. Critical regions of the response can then be identified, as shown in Fig. 3, and competitive feature sets compared on a statistical basis. It is also clear from Fig.3 that the static test only tells us that the system is operating, not whether or not the system is operational, i.e. will function as intended in the real life operational role.

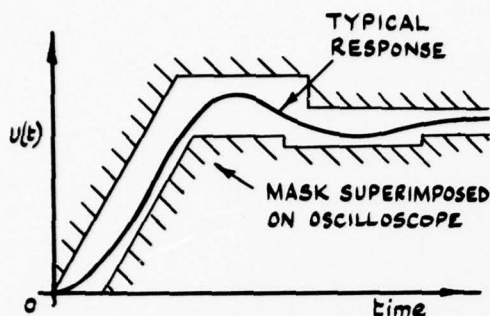


FIG. 1. MANUAL STEP TEST OF AUTOSTABILISER UNIT.

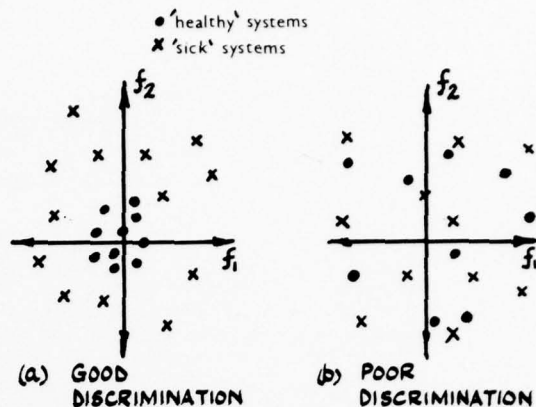


FIG. 2. SELECTION OF TEST FEATURES FOR PRE-FLIGHT CHECKOUT.

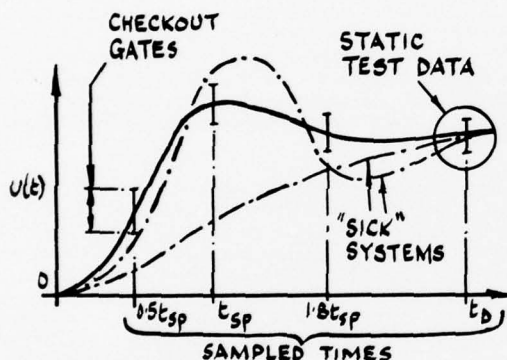


FIG. 3. SAMPLING OF AUTOSTABILISER STEP RESPONSE IN A.T.E. MODE.

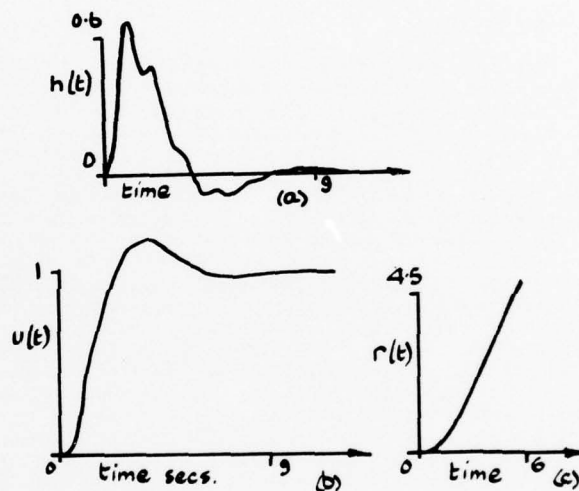


FIG. 4. ATTENUATION OF S.U.T. SECONDARY RESONANCE BY INTEGRATION.

### 3. Why Use Dynamic Testing?

The main reasons for using dynamic testing are as follows:

#### (a) User Confidence

Avionic systems are designed to respond to real time varying input signals, so that a high degree of correlation exists between mission success and the setting of suitable dynamic performance specifications. For example, in analysing the integrity of aircraft all-weather landing systems required to achieve better safety standards than one accident in  $10^7$  landings, it is possible to allocate a fraction of the permissible touchdown error to the dynamic response of the system. (3) After extensive simulator studies, it is then possible to determine a frequency response for the system which satisfies the 'false abort' case due to guidance signal noise, and to check that this design produces suitable dynamic recovery of the aircraft when disturbed by a wind gust. A dynamic performance specification for the all-weather landing system implies in turn the existence of a dynamic performance specification for all the major sub-systems written to ensure that the overall requirement is met.

#### (b) Spares Inventory Reduction

If only static performance tests are used to establish the status of a system which has to meet a dynamic operational capability, in order to confirm the integrity of the system it is found that static tolerances have to be made excessively tight. Consequently many 'healthy' systems are wrongly categorised as 'sick', resulting in setting up an excessively large spares inventory in order to provide a reasonable level of system availability. (4)

(c) Repair Costs Reduction

It follows from the previous paragraph that the repair load on the maintenance depot will be reduced because a smaller proportion of 'healthy' SUT's will be wrongly sent back for stripdown and repair. In addition, for SUT's correctly classified as 'sick', it is possible to infer from dynamic test results the possible causes of failure (5), (6), thus reducing fault locate time and hence repair costs.

(d) Increased System Reliability via Component Reduction

Dynamic tests can be designed to reduce the need for intermediate access points used to project or monitor signals needed in static test schemes. Since provision of access points degrades system reliability, their omission will be beneficial in this respect.(7)

(e) Increased System Reliability via Failure Prediction

It has been stated that for integrity analysis of the Concorde, autopilot failures may reasonably be categorised as having equal probability for 'catastrophic' changes and 'drift' changes in performance (8). Although as yet there is no direct evidence that dynamic testing may help in predicting impending 'catastrophic' changes in performance, it has been suggested that prediction of gradual degradation via regular testing and time series analysis is feasible, (9) so that potentially 'sick' SUT's may be removed prior to failure and restored to a satisfactory condition.

4. Transfer Function Models

For linear systems, a transfer function model of the form,

$$H(s) = \frac{\sum_{i=0}^{i=q} b_i s^i}{\sum_{i=0}^{i=n} a_i s^i} \quad [1]$$

where  $s$  is the Laplace operator, defines the response of the SUT for all possible inputs. For a test stimulus  $X(t)$  with Laplace transform  $X(s)$ , the response of the SUT may be written

$$Y(t) = \mathcal{L}^{-1} X(s) H(s) \quad [2]$$

Of particular interest in pre-flight dynamic testing are the following standard responses:-

(a) the impulse response (or weighting function) defined by

$$h(t) = \mathcal{L}^{-1} H(s)$$

(b) the previously met step response, defined by,

$$u(t) = \mathcal{L}^{-1} \frac{H(s)}{s}$$

and (c) the ramp response, defined by,

$$r(t) = \mathcal{L}^{-1} \frac{H(s)}{s^2}$$

Of these responses, the step response is the one most commonly used in avionic systems and is readily available in standard form for a wide variety of transfer functions.(10) It should be noted that if the system is mildly non-linear, which is usually the case with present day design and manufacturing skill levels, the transfer function model may still be a satisfactory representation of the SUT under the stipulated test conditions, but care is then needed in interpretation of the results to other test situations.(11)

The popularity of the step input is partly due to the simplicity of signal generation and partly to the intuitive understanding of this response by system designers.(12) Direct impulse testing is not favoured due to the problems of signal generation and excessive disturbance of the SUT, but as we shall see later, the impulse response may be estimated indirectly via pseudo-noise sequence injection and output-input cross-correlation which overcomes these difficulties. Ramp testing is frequently used in tracking systems since the ramp function may be regarded as similar to operational inputs often experienced during at least part of the mission. The steady state ramp error occurring after 'lock-on' is then of particular interest.

Although in theory the impulse, step, and ramp responses contain the same information on system performance, in practice the extraction of the information can be made difficult by an unsatisfactory choice of test stimulus. This is particularly true if the steady state ramp error is inferred from the SUT step response, since any integration inaccuracies will affect the final estimate of a relatively small quantity which is often regarded as of fundamental importance. When estimated from the ramp response, the steady state error is, of course, obtained from a single measurement. A less obvious, but equally useful observation on test stimulus selection concerns the amplification of SUT secondary resonances via impulse testing (there will be applications where secondary resonances due to drive mechanisms, structural deflections etc. will need to be assessed as part of the checkout test, and other applications where this need not be done). To see why this is so, we write equation [1] into SUT pole-zero form,



$$H(s) = \frac{\prod_{i=1}^k (1 + T_i s) \prod_{i=q}^i (1 + \frac{2\zeta_{is}}{\omega_{ni}} + \frac{s^2}{\omega_{ni}^2})}{\prod_{j=1}^n (1 + T_j s) \prod_{j=n}^j (1 + \frac{2\zeta_{js}}{\omega_{nj}} + \frac{s^2}{\omega_{nj}^2})} \quad [3]$$

where  $\prod$  is the product sign. We do not commit ourselves at this stage on the specific breakdown between real and complex factors in equation [3], so some of the limits are left open. The SUT zeros are given by

$$\left. \begin{aligned} z_i &= -1/T_i & (\text{for real } z) \\ z_{i,i+1} &= -\zeta_{\omega_{ni}} \pm j\omega_{ni} \sqrt{1 - \zeta_i^2} & (\text{for complex } z) \end{aligned} \right\} \quad [4]$$

and the SUT poles (or 'modes') are given by

$$\left. \begin{aligned} p_j &= -1/T_j & (\text{for real } p) \\ p_{j,j+1} &= -\zeta_{\omega_{nj}} \pm j\omega_{nj} \sqrt{1 - \zeta_j^2} & (\text{for complex } p) \end{aligned} \right\} \quad [5]$$

The solution for  $Y(t)$  may now be written in terms of the residues  $A_j$  (of which  $\phi_j$  is a constituent for complex modes), as would be obtained on solution of equation 2 as follows,

$$Y(t) = \sum_{j=1} A_j e^{-s/T_j} + \sum_{j=n}^j A_j e^{-\zeta_j \omega_{nj} t} \sin(\omega_{nj} \sqrt{1 - \zeta_j^2} t + \phi_j) + \text{steady state terms} \quad [6]$$

The residues depend on the SUT poles and zeros, but also on  $X(t)$ , and it is their dependence on  $X(t)$  which can be turned to advantage in system testing.

As an example, consider the case of a fourth order SUT, in which there are two complex modes of damping ratio  $\zeta_1$  and  $\zeta_2$  respectively,  $\zeta_2$  typically being lightly damped. In the  $s$  plane, the frequency separation is  $(\omega_{n2}/\omega_{n1}) = \lambda$ , where  $\lambda \gg 1$ , by definition of a secondary resonance. If  $A_1$  and  $A_2$  are the residues at the system poles, it is readily shown that (13)

$$\left| \frac{A_1/A_2}{A_1/A_2} \right| \frac{h(t)}{u(t)} = \lambda \quad [7]$$

The ramp residues are similarly attenuated with respect to the step response by a factor  $\lambda$ . Consequently, the secondary mode, even when lightly damped, is heavily attenuated each time the test stimulus is integrated, as shown in Fig.4. Therefore one advantage of the impulse-like test is the exposure to view of the higher resonances so that the system designer is forced to consider their implication (if any) on operational performance. Although the theory behind equation 7 is based on a fourth order transfer function, these attenuation effects are present irrespective of the order of the system, as confirmed in the testing of a high order SUT. (14)

##### 5. Selection of Checkout Features for Step Testing

It is clear from section 4 that high frequency modes are generally severely attenuated in the system step response, so that the decision to implement such a checkout test implies that such secondary modes are (a) not present, (b) are unimportant, or (c) are checked via a separate test. As a consequence, the selection of suitable test features based on sampled values of the step response is simplified, so that even if a simulation study is implemented to find a 'best' set of features as suggested in section 2, it should be possible to start with a near optimum solution.

Based on a selection of studies on various SUT's the present author suggests that four sample times should prove satisfactory for pre-flight checkout, since these time delays are generally sensitive (as a set) to parameter changes). These are shown in Fig.3.

$$[F_s]^T = [u(0.5t_{sp}); u(t_{sp}); u(1.8t_{sp}); u(t_D)]^T \quad [8]$$

where  $t_{sp}$  is the time to first peak overshoot of the nominal SUT, and  $t_D$  is the practical SUT decay time,  $[F_s]^T$  is shorthand notation for the feature vector (set of measurements) used in checkout decision making. Even if the test designer is not operating under constraints imposed by system test time or computer capacity, the discrete feature vector should not be made needlessly long, since there is a danger of factors important to the operational efficiency of the SUT are not submerged in a wealth of unimportant detail.

Ideally, the gate widths and test feature selection need confirmation from field trials or simulation studies before final committal within the test schedule. One technique for initial gate width selection is to assign realistic tolerances to all parameters, and then compute the expected boundaries of



performance variation of 'healthy' systems using sensitivity functions and assuming the parameter independence rule to hold. (15) Hence if  $\alpha_j$  is the  $j$ th parameter, then the gate width  $\pm g_i$  is given by,

$$\pm g_i = \left\{ \sum_{j=1}^{j=J} \left( \frac{\partial F_i}{\partial \alpha_j / \alpha_j} \right)^2 \left( \frac{\Delta \alpha_j}{\alpha_j} \right)^2 \right\}^{1/2} \quad [9]$$

where  $\frac{\partial F_i}{\partial \alpha_j / \alpha_j}$  are sensitivity functions, and  $\Delta \alpha_j / \alpha_j$  is the % tolerance set on the  $j$ th parameter, there

being  $J$  parameters in all.

#### 6. Selection of Checkout Features for Impulse Testing

If high frequency secondary oscillatory modes are not observable on the impulse response, feature selection for checkout is straightforward, and in like manner to section 5, a feature vector with four elements will often prove adequate. A reasonable set for initial investigation is,

$$[F_i]^T = [h(t_{IP}); h(2t_{IP}); h(3t_{IP}); h(t_D)]^T \quad [10]$$

where  $t_{IP}$  is the time to impulse response peak of the nominal SUT. When high frequency lightly damped secondary modes are observable such as in Fig.4(a) the problem becomes much more complicated because acceptable changes in the dominant mode can render gates set to constrain the secondary mode ineffective. A possible solution is to supplement the feature vector of equation 10 with a further three features chosen close together and within the first observable period of secondary oscillation.

It should be noted that unless there are many dynamic performance requirements written into the SUT operational specification, it is extremely unlikely that the number of test features needed for pre-flight checkout will approach the  $(n + q + 1)$  minimum needed for transfer function identification because the information on whether or not the system is operational is contained in just a few measurements. The needs of testing for pre-flight checkout and design proving are therefore significantly different and this fact is already exploited by test designers since it is known that many existing test schedules call for as few as six data points.

#### 7. System Frequency Response

If a stable linear system with transfer function  $H(s)$  is excited by a sinusoidal signal  $a \sin \omega t$ , it is well known that after an initial transient phase, the system will settle down to a steady sinusoidal response of the same frequency as the input. In general there will be a phase shift  $\phi$  and the output waveform will be of a different amplitude to the forcing function, so that in the absence of measurement noise the output signal may be written as  $ka \sin(\omega t + \phi)$ . It is also readily shown that the solution of the differential equation describing the steady state behaviour is obtained by substituting  $j\omega$  for  $s$  in equation [1]. This gives a rotating vector

$$H(j\omega) = |H(j\omega)| e^{j\phi} \quad [11]$$

which after writing in the form  $\frac{A + jB}{C + jD}$  can be put in standard polar notation as

$$\left. \begin{aligned} |H(j\omega)| &= \sqrt{\frac{A^2 + B^2}{C^2 + D^2}} \\ \phi &= \tan^{-1}(B/A) - \tan^{-1}(D/C) \end{aligned} \right\} \quad [12]$$

Equation [12] gives the information available from the steady state response to a single sinusoidal input,  $\phi$  being the aforementioned phase shift and  $k = |H(j\omega)|$  being the system amplitude ratio. Because both phase and amplitude ratio are available we have two test features per test frequency. It is important to make use of both features if test time needs to be reduced. In calculating the theoretical system frequency response, discrete values of  $\omega$  are substituted into  $H(j\omega)$  and the results plotted as a function of  $\omega$ .

If an individual sine wave is injected into the SUT, one amplitude ratio and one phase estimate is made available, so that if the test is repeated, a series of discrete points may be plotted in exactly the same manner as for the calculated values. This is the serial method of test, in which we wait until the system has settled at each test frequency before a measurement is made. Total test time is therefore some non-linear function of settling time multiplied by the number of test frequencies needed. At the design and development stage of a system, a wide ranging frequency response plot such as shown in Fig.5 is essential for proving the design, and for finger-printing purposes. Test time for such wideband information can be considerably reduced still using an essentially serial mode technique in which the sinusoidal input has a slowly time varying frequency of excitation, (16) and this has proved an extremely useful method in the past even when using such crude displays of the return signal as chart recorders.

The slow sweep method is based on the fact that by careful choice of frequency sweep characteristics, the envelope of the system output, when plotted as a function of time, approximates to the system amplitude ratio, and the phase shift may be recovered as well. With such a slow sweep technique a satisfactory test time for a second order system with natural frequency  $\omega_n$  r/s would be of the order of  $100/\omega_n$  seconds.

The slow sweep frequency may be obtained using special purpose instruments, or from a computer controlled Fourier Response Analyser (FRA) of the type to be described in the next section.

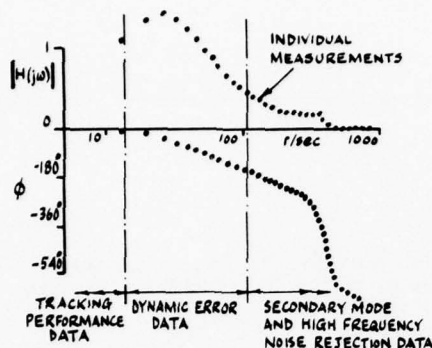


FIG. 5. FREQUENCY RESPONSE OF S.U.T.

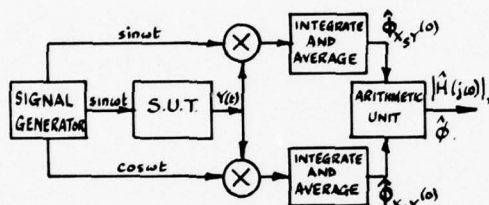
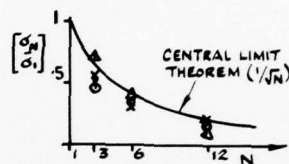


FIG. 6. F.R.A. PRINCIPLE.



(a) RETURN SIGNAL TO F.R.A.



Δ  $|H(jω)|$  @ 55 Hz  
 x  $φ$  @ 55 Hz  
 •  $|H(jω)|$  @ 66 Hz  
 ○  $φ$  @ 66 Hz  
 (b) REDUCING SPREAD BY MEASURING OVER 'N' CYCLES

FIG. 7. APPLICATION OF FRA TO NOISY S.U.T.

## 8. Correlation Techniques in Frequency Response Measurement

The serial mode frequency response method of system testing is now well established on a world-wide basis, and many pre-flight checkout applications have been reported. As far back as 1961, the method was a recommended test for USAF equipment, (17) and has mushroomed ever since. Although the method may be implemented in a wide variety of ways, (18) including the use of oscillators, phase variable filters and oscilloscope displays, in a manual test mode, and sampling plus counting techniques in digital computer aided test stations, the most universally used technique involves correlation, which is needed to reduce the effect of measurement noise on gain and phase estimates. The frequency domain test instrument designed to exploit the correlation principle has become known as the Fourier Response Analyser (FRA) because gain and phase estimators are identical to the Fourier series coefficients used to describe a repetitive waveform.

By definition, the cross-correlation function  $φ_{XY}(τ)$  of two signals  $X(t)$  and  $Y(t)$  is,

$$φ_{XY}(τ) = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^{+T} X(t - τ)Y(t) dt \quad [13]$$

For sinusoidal excitation we have,

$$X_s(t) = a \sin ωt \quad [14]$$

with a noisy return signal from the SUT of,

$$Y(t) = k.a. \sin(ωt + φ) + n(t) \quad [15]$$

We also require the cosine signal,

$$X_c(t) = a \cos ωt \quad [16]$$

to be available as well, which is readily achieved using a slave oscillator or a time delay unit. It is implicit that the integration time,  $T$ , is chosen to be an integer multiple of the input sine wave, so that  $T = 2πN/ω$ . We now correlate the return signal (equation [15]) with the sine and cosine signals of equation [14] and [16] respectively, and set  $τ = 0$ . (19) The result is

$$φ_{X_s Y}(0) = \frac{ka^2}{2} \cos φ + \frac{a}{T} \int_0^T n(t) \sin ωt dt \quad [17]$$

which is the in-phase estimate,

and

$$\phi_{X_c Y}(0) = \frac{ka^2}{2} \sin \phi + \frac{a}{T} \int_0^T n(t) \cos \omega t dt \quad [18]$$

which is the quadrature estimate.

On the assumption that noise may be neglected, the gain and phase estimators become,

$$\hat{k} = \frac{2}{a^2} \{ [\phi_{X_s Y}(0)]^2 + [\phi_{X_c Y}(0)]^2 \}^{1/2} \quad [19]$$

which is, of course, the estimate of  $|H(j\omega)|$ ,

and

$$\hat{\phi} = \tan^{-1} \left[ \frac{\phi_{X_c Y}(0)}{\phi_{X_s Y}(0)} \right] \quad [20]$$

A block diagram of the FRA principle is shown in Fig.6. Having been available commercially in analogue form for many years, it is now available from several manufacturers in digital form with computer control capability for use in an automatic test set.

The correlation process may be regarded as a filter through which we can observe a cleaned up version of a noisy return signal, thereby providing reasonable estimates of  $k$  and  $\phi$ . An important feature of the FRA is the ability to reject harmonics present in the return signal, since,

$$\left. \begin{aligned} \frac{\omega}{2\pi N} \int_0^{2\pi N/\omega} \sin(n\omega t + \phi_n) \sin \omega t dt &= 0 \\ \frac{\omega}{2N} \int_0^{2\pi N/\omega} \sin(n\omega t + \phi_n) \cos \omega t dt &= 0 \end{aligned} \right\} \quad [21]$$

(for  $n = 2, 3, 4, \dots$ )

At non-harmonic noise frequencies, the errors in  $\phi_{X_s Y}(0)$  and  $\phi_{X_c Y}(0)$  may be calculated from theoretical considerations (20) as can the effect of white noise on the in-phase and quadrature measurements (21). The effect of sinusoidal and white noise on gain and phase estimates  $k$  and  $\phi$  is much more difficult to predict. One fact which does emerge strongly, however, is that measurement variance is reduced as measurement time is increased, so that  $N$  is chosen to achieve adequate noise rejection. In field work the present author has found the central limit theorem ( $\sigma$  proportional to  $1/\sqrt{N}$ ) to be a reasonable guide to measurement time selection, (22) as shown in the example of Fig.7, which also illustrates the point that although correlation greatly reduces the effect of noise, in the practical situation perfect filtering is unlikely to be achieved.

#### 9. Selection of Checkout Features for Frequency Response

If the serial mode frequency domain technique is used in situations of limited SUT test time, it becomes extremely important to select a few frequencies which will yield the necessary confidence in the operational status of the system. Fortunately, in the frequency domain, this is not difficult, because the operational function of the SUT partitions nicely into the following three regions (22) as illustrated in Fig.5.

(a) low-frequency region where the SUT generally is expected to track the input closely. In order to obtain the necessary resolution it is useful in a feedback system to monitor the error signal directly rather than estimate the error from input and output measurements. The information yielded from the low frequency region relates to time domain data in the decay time region, and can usually be obtained from one test frequency at about 5% bandwidth.

(b) mid-frequency region encompassing peak amplitude ratio ( $M_p$ ) and bandwidth ( $\omega_b$ ). This region primarily determines such important time domain performance characteristics, as  $I_p$ ,  $S_p$ ,  $R_p$  (ramp peak), and the times at which these peaks occur, proving that on a practical basis, time and frequency domain test methods are interchangeable. These time domain criteria can be adequately constrained by three test frequencies  $f_{450}$ ,  $f_{900}$  and  $f_{1350}$  chosen on the basis of SUT nominal performance so that

$$[F_i]^T = [ |H(j\omega)|_{f_{45}} ; \phi_{f_{45}} ; |H(j\omega)|_{f_{90}} ; \phi_{f_{90}} ; |H(j\omega)|_{f_{135}} ; \phi_{f_{135}} ]^T \quad [22]$$

The use of three such test frequencies for pre-flight SUT checkout is very popular since it is an adequate

feature vector for many SUT's. In addition to FRA and digital computation methods, analogue based BITE (built-in-test-equipment) designed only to test at such selected frequencies is also used.

(c) high frequency region well beyond bandwidth, encompassing noise rejection and secondary resonance requirements. In the time domain, the high frequency noise rejection is compressed into the region around  $t = 0$ , and the secondary modes are superimposed in the manner already seen in Fig.4, but in the frequency domain the secondary mode of Fig.5 is well separated from the dominant mode, and can be detected for checkout purposes by three further test frequencies, one at the nominal value of modal frequency, plus one either side. (14) High frequency noise rejection beyond bandwidth can be ascertained from one or two test frequencies depending on the rate of roll-off sought, so that an extensive test schedule may appear as shown in Table I.

Table I

Test Frequency (in Terms of Nominal SUT Response)	$f_{50^\circ}$	$f_{45^\circ}$	$f_{90^\circ}$	$f_{135^\circ}$	$f_{180^\circ}$	$f_{260^\circ}$	$f_{290^\circ}$	$f_{340^\circ}$
Transfer Function Measured	$\theta/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$	$\theta_o/\theta_i$
Purpose of Test	Tracking Performance Check	Dynamic Performance Check			Noise Rejection Check	Secondary Mode Check		

#### Derivation of Frequency Domain Test Schedule for Complex SUT

We therefore conclude that frequency domain feature selection relates naturally to the performance specification for the SUT, and to the design procedure as well, since classical frequency domain techniques remain popular for this purpose. If dynamic errors are the only performance aspects needing checkout, as is frequently the case, then three test frequencies are sufficient. Noise rejection and tracking performance each require at least one more test frequency. A secondary resonance requires three further test frequencies, but unless the measurement noise is severe the test time is not excessive. If more than one secondary resonance is important either the swept frequency method or the spectral analysis method to be discussed later is suggested.

#### 10. Indirect Impulse Testing Via PNS Excitation and Cross-Correlation

##### 10.1 Development of the Convolution Integral in Terms of Input Autocorrelation Function

In recent years the pulse testing time domain and serial mode frequency domain techniques have been rivalled by the appearance of pseudo-noise test signals, which for linear systems, can, via the cross-correlation principle, yield under specified conditions a realistic approximation to the system impulse response without the physical injection of an impulse stimulus. (23)(24) The theoretical basis for this work is the convolution integral, so that if  $X(t)$  and  $Y(t)$  are the SUT input and output signals at time  $t$ , and  $h(t)$  is the SUT impulse response as before, then

$$Y(t) = \int_{-\infty}^{\infty} h(\tau) X(t - \tau) d\tau \quad [23]$$

If we now combine equation 23. (convolution) with equation 13. (cross-correlation), we may write,

$$\phi_{XY}(\tau) = \frac{1}{T} \int_0^T X(t - \tau_1) dt \int_0^T h(\tau_1) X(t - \tau_1) d\tau_1 \quad [24]$$

interchanging the order of integration we have,



$$\phi_{XY}(\tau) = \frac{1}{T} \int_0^T h(\tau_1) \int_0^T X(t - \tau_1) X(t - \tau_1) dt d\tau_1$$

which can be written in final form as,

$$\phi_{XY}(\tau) = \int_0^T h(\tau_1) \phi_{XX}(\tau - \tau_1) d\tau_1 \quad [25]$$

where  $\phi_{XX}(\tau)$  is the autocorrelation function of the input signal averaged over measurement time  $T$ . Equation 25 is the Wiener-Hopf equation, and in particular, if  $\phi_{XX}(\tau)$  is the unit impulse, then  $\phi_{XY}(\tau) = h(\tau)$  and equation 25 is an impulse response estimator for the SUT.



FIG. 8. REMOVING UNCERTAINTY IN  $\phi_{XY}(\tau)$  BY USING P.N.S. STIMULUS.

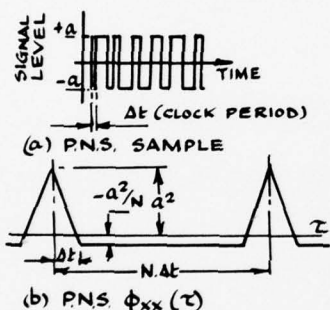


FIG. 9. P.N.S. CHARACTERISTICS.

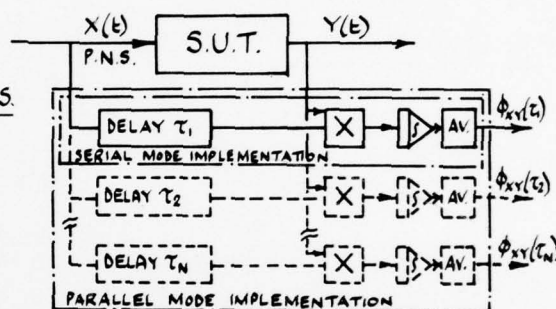


FIG. 10. IMPLEMENTATION OF IMPULSE TESTING VIA P.N.S. INJECTION AND CROSS CORRELATION.

### 10.2 P.N.S. Characteristics

White noise has the unit impulse autocorrelation function required by equation 25 but unfortunately infinite measurement time is implied for satisfactory estimates. As can be seen from Fig. 8, the uncertainty due to the test signal can be removed by using a pseudo-noise sequence (PNS) with precisely defined statistical properties which sufficiently approximate to white noise for the purpose of dynamic testing. (24) Two-level sequences (PRBS), are particularly attractive for this purpose since they are easily generated by shift registers incorporating the necessary feedback and operating on modulo 2 arithmetic. The resultant test signal and autocorrelation function are shown in Fig. 9, and the schematic mechanisation is shown in Fig. 10, the method of test being either serial or parallel mode depending either on the number of delay lines provided, or on the provision of intermediate storage prior to computation.  $\phi_{XX}(\tau)$  is a function of the clock period  $\Delta T$ , and the amplitude of the pulse  $\pm a$ , which may be conveniently expressed as follows: (25)

$$\phi_{XX}(\tau) = \frac{(N+1)}{N} a^2 \Delta t \delta(\tau) - \frac{a^2}{N} \quad [26]$$

where  $\delta(\tau)$  is the Dirac delta function. The 'spike' in equation 26 repeats with a periodicity  $N \Delta t$ , where  $N = (2^R - 1)$ ,  $R$  being the length of the generator shift register.  $R = 10$  is a common length, giving a test sequence periodicity of  $1023 \Delta t$  but success with a specific range of SUT's has been achieved with  $R$  as low as 6, giving  $N = 63$  bits. As  $R$  increases, so is the error in estimation of  $h(t)$  due to d.c. offset reduced, and the need for post-test correction avoided. However, this is a minor reason in the choice of  $N$ , since post-test correction is not difficult, and the problem may be avoided altogether by the use of inverse repeat sequences.

### 10.3 Matching the PNS to the SUT

If the system delay time is less than  $N \Delta t$ , the cross-correlation function for the SUT excited by two-level PNS is, (25)

$$\phi_{XY}(\tau) = \frac{a^2(N+1)\Delta t}{N} h(\tau) - \frac{a^2}{N} \int_0^{N\Delta t} h(t) dt \quad [27]$$

and it is clearly useful to reduce or eliminate the second term on the right hand side of equation [27] by making  $N$  large, or by using inverse repeat sequences. However, in selecting PNS characteristics for system testing, it is useful to place the graphical interpretation of Fig.11 on equation [25] so that the

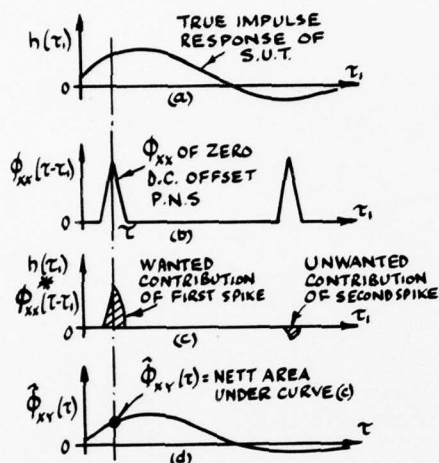
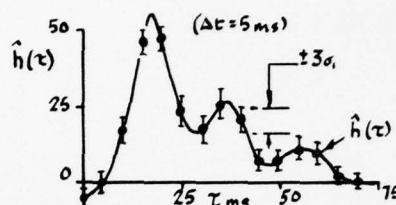
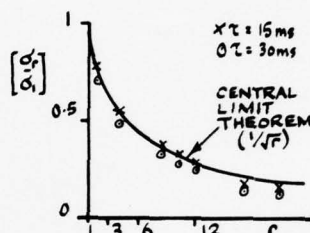


FIG. 11 GRAPHICAL INTERPRETATION OF WIENER-HOPF EQUATION AS AN AID TO P.N.S. SELECTION.



(a) ESTIMATED  $\hat{h}(t)$  FOR S.U.T. OF FIG. 5. USING P.N.S. EXCITATION AND CORRELATION.



(b) REDUCING MEASUREMENT SPREAD BY AVERAGING OVER 'r' SEQUENCES.

FIG. 12. P.N.S. TESTS ON NOISY S.U.T.

various sources of error other than d.c. offset may be understood. These errors can be dealt with in three parts.

(a) equation [25] implies integration over all time: in the practical situation this means that the product of the two functions must be zero outside the time span of integration, which in turn requires the second spike to occur after the response is over. The PNS sequence length must therefore be somewhat longer than the decay time of the system

(b) the initial value will, in general, be in error, because the triangular autocorrelation function is centred at the origin, so that the product will differ from the true impulse response. Even for a narrow autocorrelation function, there will be an error between the actual and estimated impulse response as can be seen in Fig. 8. This particular source of error disappears for  $\tau > \Delta t$ .

(c) errors due to the finite width of the autocorrelation function clearly depend on the behaviour of  $h(t)$  in any time interval  $2\Delta t$ . From Fig. 11 it is clear that distortions can take place in regions of high rates of change of  $h(t)$ . In particular, oscillations present on the impulse response which have a period comparable with the pulse width are removed by the multiplying and integrating action of convolution as will be seen in section 10.5. As would also be expected from an information theory approach,  $\Delta t$  must be chosen by considering the highest frequency likely to be present in the SUT impulse response, and the following analysis has proved helpful in making the choice.

In a theoretical study undertaken to detect oscillatory modes, it has been shown that in order to detect the peak to within 1%, the ratio of (modal period/clock period) must be about 20:1. (26) As this ratio decreases, the accuracy of estimation falls off rapidly, as shown in Table II, a ratio of 10:1 appearing to be a reasonable compromise choice for good resolution. The existence of secondary modes can, of course, be detected with a much lower ratio of (modal period/clock period) than 10:1, as a number of case studies have shown, but the secondary mode is then greatly attenuated compared to the true impulse behaviour.

TABLE II Accuracy of Detection of Sine Wave Amplitude Using PNS and Cross Correlation (26)

$\left( \frac{\text{Resonance Period}}{\text{PNS Clock Period}} \right)$	$\left( \frac{\text{Estimated Peak Amplitude}}{\text{True Peak Amplitude}} \right)$
10.00	97.5%
3.33	74%
2.00	41%
1.43	14%
1.10	1½%

AD-A041 042

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT--ETC F/G 1/3  
INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS.(U)  
1977

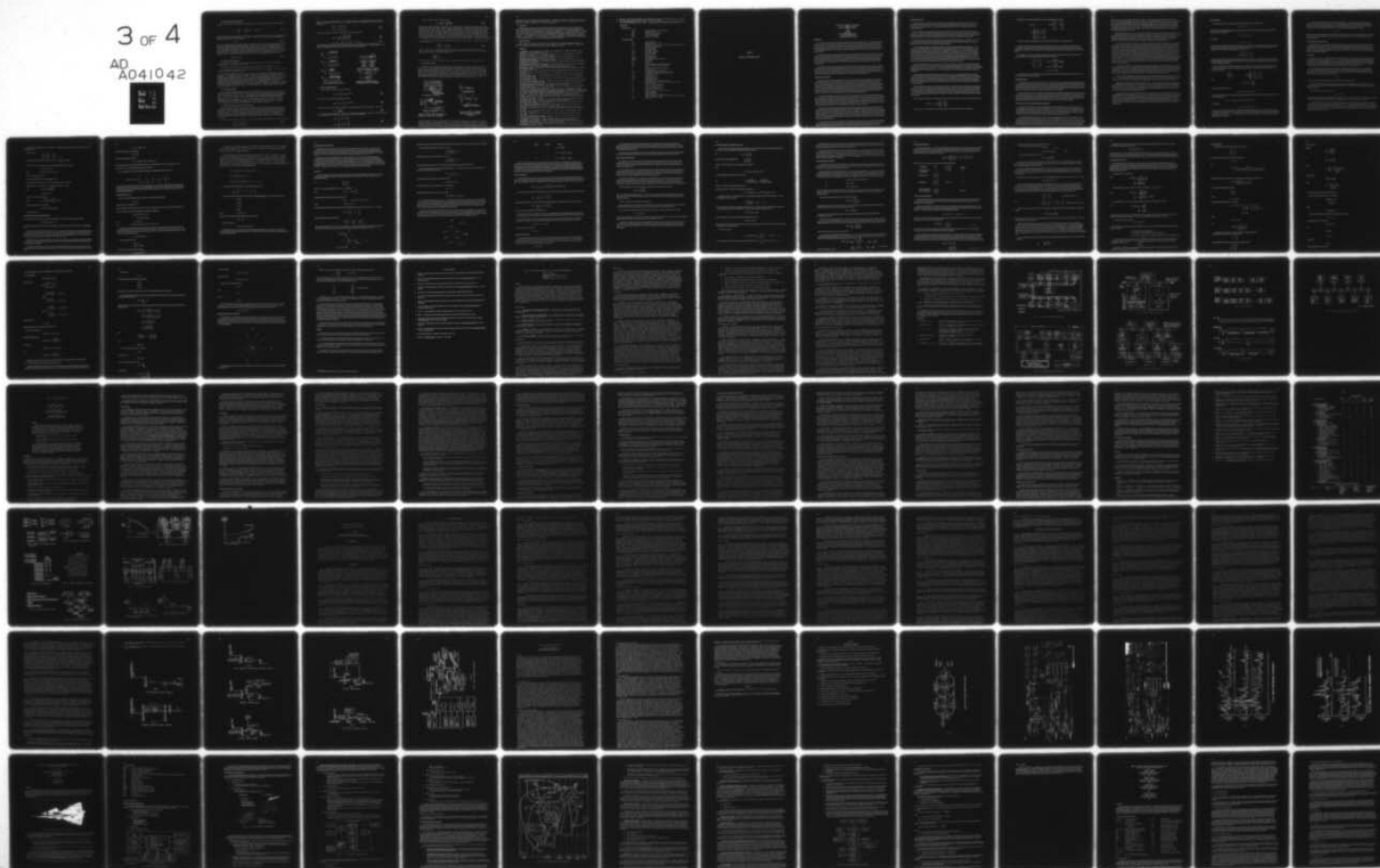
UNCLASSIFIED

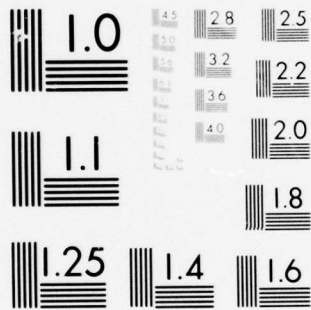
AGARD-OGRAPH-224

NL

3 OF 4

AD  
A041042





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A



#### 10.4 Noise Rejection Characteristics

The error variance in estimating SUT impulse response via PNS and cross-correlation may be expressed as, (21)

$$\sigma^2 = \frac{1}{T^2} \int_0^T dt \int_0^T dt^1 \phi_{nn}(t^1 - t) \phi_{XX}(t^1 - t) \quad [28]$$

provided the noise and test signals are uncorrelated. If the corrupting noise is white, then  $\phi_{nn}(t) = \rho \delta(t)$ , where  $\rho$  is the spectral density, and  $\delta(t)$  is the Dirac delta function. Substituting in equation [28] and integrating results in the simple expression

$$\sigma = a \sqrt{\frac{\rho}{T}} \quad [29]$$

so that the standard deviation is proportional to the signal level  $\pm a$ , and is inversely proportional to the square root of measurement time divided by spectral density, i.e. the central limit theorem applies. A further interesting conclusion from equation [29] is that the variance error is independent of delay time. Fig.12 shows the cross-correlation function for the SUT previously studied in Fig.5 and measured in a typical maintenance environment of 'a priori' indeterminate noise level and it can be seen that the variance error is reasonably constant and is reduced by approximately  $(1/\sqrt{r})$  if correlation takes place over  $r$  sequences of PNS. If required equation 29 may also be expressed as a signal-to-noise ratio. Since the equivalent impulse generated by the PNS test signal is at  $a^2 \Delta t$ , the signal-to-noise ratio will be:-

$$\frac{S}{N} = h(t) a \Delta t \sqrt{\frac{T}{\rho}} \quad [30]$$

#### 10.5 Example on PNS Selection

Suppose we wish to identify the SUT of Fig.4(a) using PNS and cross-correlation to estimate the impulse response. The procedure is as follows,

(a) From the observed decay time of the complete impulse response of 12 seconds,  $N \Delta t > 12$ .

(b) From accuracy considerations in identifying the secondary mode,  $\Delta t < \frac{2\pi}{5 \times 10^3}$ , if we choose 10 clock intervals per period of the secondary mode.

If  $\Delta t$  is made 0.05Hz then  $N > 240$ , so that  $R = 8$  giving  $N = 253$  would be satisfactory for accurate identification. The practical effect of varying  $\Delta t$  and  $N$  is shown in Fig.13 and confirms the chosen PNS parameters. The remaining variables in the test schedule,  $a$  and  $r$ , determine the accuracy of the estimates given a satisfactory choice of  $\Delta t$  and  $N$ . This can be done by assuming a value of  $\rho$  and choosing a desired signal-to-noise ratio, but is better done when typical measurement noise characteristics for a family of SUT's becomes available, since in the present author's experience the noise level can vary considerably within a batch of apparently similar SUT's. For the example shown in Fig.12(a),  $r = 1$  is considered too short a measurement time since the uncertainty band is comparable to the permissible variation in performance within the 'healthy' family. If the  $\pm 3\sigma$  bounds due to noise are to be kept within  $\pm 2$  units (i.e.  $\pm 4\%$  of  $I_p$  which is  $\approx 50$  units) which is more reasonable, then  $r = 4$  would appear to be a suitable choice, since  $\pm 3\sigma_1 = 4$  units.

#### 10.6 Test Time Using PNS

For correlation over  $r$  sequence lengths, total correlation time  $T$  in equation 25. =  $r N \Delta t$ . It is customary to allow one complete sequence of PNS to 'initialise' the SUT prior to correlation commencing, so the total test time is  $(r + 1) N \Delta t$ . In the early days of PNS testing via special-to-type instruments, only one delay line was available, so that a test time of  $(r + 1) N \Delta t$  was required per each point on  $\phi_{XY}(\tau)$ . More recent instruments have typically provided 100 delay lines, so that 100 points on  $\phi_{XY}(\tau)$  can be estimated from a test time  $(r + 1) N \Delta t$  in the so-called 'parallel' mode of Fig.10. The PNS technique can also be implemented directly by digital computer, but care must be taken to design the test schedule so as not to exceed the computer capacity, (1) although as seen in section 6, this need not be a handicap in pre-flight testing since only a few test features are needed.

#### 10.7 Frequency Response Directly from PNS Injection

PNS stimuli have a precisely defined frequency domain  $(\sin x/x)^2$  line spectrum with spectral lines occurring at  $(2\pi/N\Delta t)$ ;  $(4\pi/N\Delta t)$ ; etc. with nodes occurring at integer multiples of the clock frequency  $(2\pi/\Delta t)/\text{sec}$ . Because the input spectrum is so precisely defined, PNS may be regarded as a parallel-mode frequency stimulus. Although the frequency response could be obtained from Fourier transforming  $\phi_{XY}(\tau)$  in the normal way, if only a few frequency data points are required for checkout as suggested in section 9, significant reductions in computing time result from taking advantage of the known spectral characteristics, which have been tabulated over the frequency range of interest. (26) The approach is as follows:

At the  $r^{\text{th}}$  spectral line (at frequency  $\omega_r$ ), the in-phase and quadrature components of the PNS input are,

$$\left. \begin{aligned} R_i(\omega_r) &= A_{ri} \cos \phi_{ri} \\ Q_i(\omega_r) &= A_{ri} \sin \phi_{ri} \end{aligned} \right\} \quad [31]$$

where  $A_{ri}$  and  $\phi_{ri}$  are known 'a priori'. If the return signal is correlated with  $\sin(\omega_r)$  and  $\cos(\omega_r)$  in turn, as outlined in section (8), then the in-phase and quadrature components of the SUT output are estimated as,

$$\left. \begin{aligned} R_o(\omega_r) &= A_{ro} \cos \phi_{ro} \\ Q_o(\omega_r) &= A_{ro} \sin \phi_{ro} \end{aligned} \right\} \quad [32]$$

the required checkout data  $|H(j\omega_r)|$  and  $\phi_{\omega_r}$  can now be evaluated from,

$$|H(j\omega_r)| = \frac{A_{ro}}{A_{ri}} = \sqrt{\frac{R_o^2 + Q_o^2}{R_i^2 + Q_i^2}} \quad [33]$$

$$\phi_{\omega_r} = [\phi_{ro} - \phi_{ri}] = \tan^{-1}(Q_o/R_o) - \tan^{-1}(Q_i/R_i) \quad [34]$$

As an example of the benefit obtained using this approach, if the FFT algorithm is incorporated, a reduction in computation time of 30:1 is estimated if only three frequencies are required compared to the method of obtaining the complete cross-correlation function  $\phi_{xy}(\tau)$  first.

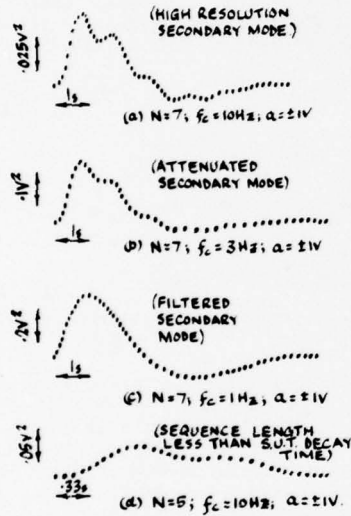


FIG 13. MATCHING PMS TO SUT.

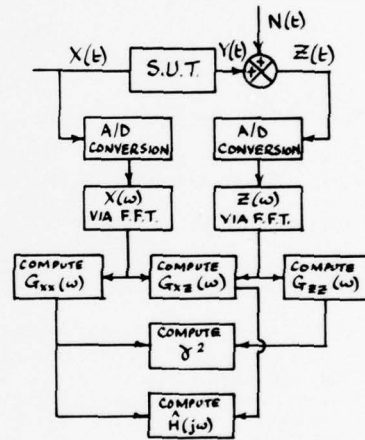


FIG 14. BLOCK SCHEMATIC OF SPECTRAL ANALYSIS TECHNIQUE.

#### 11. Spectral Analysis Methods

The auto spectral density  $S_{xx}(\omega)$  is defined by

$$S_{xx}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi_{xx}(\tau) e^{-j\omega\tau} d\tau \quad [35]$$

and the cross spectral density similarly defined by

$$S_{xy}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi_{xy}(\tau) e^{-j\omega\tau} d\tau \quad [36]$$

and until recently system frequency response has been determined via these relationships, (28) since

$$H(j\omega) = \frac{S_{xy}(\omega)}{S_{xx}(\omega)} \quad [37]$$

is a frequency response estimator. However, it is generally preferable to compute  $H(j\omega)$  directly from the Fourier transform of the input and output signals using the relationships,

$$\left. \begin{aligned} X(\omega) &= \int_{-\infty}^{\infty} X(t) e^{-j\omega t} dt \\ Y(\omega) &= \int_{-\infty}^{\infty} Y(t) e^{-j\omega t} dt \end{aligned} \right\} \quad [38]$$

The SUT transfer function is then given by

$$H(j\omega) = \frac{G_{yx}(\omega)}{G_{xx}(\omega)} = \frac{Y(\omega) X^*(\omega)}{X(\omega) X^*(\omega)} \quad [39]$$

where \* means complex conjugate. In practice, the calculations will be performed on discrete data, and the resultant implications, and a detailed proof of the equivalence of the direct method with the correlation method are to be found in reference (29). Computer implementation as suggested for testing electronic SUT's is shown in Fig.14, (30), the Fast Fourier Transform (FFT) being used in view of the enormous reduction in computing effort thereby achieved. Reference (30) presents a number of broad spectrum frequency response results similar to Fig.5 but obtained using spectral analysis methods although no recommended input stimulus is given in that paper. In addition to PNS already suggested herein as a test stimulus, white noise (31) and a fast frequency sweep (32) have been used as test signals in spectral analysis. The method has been also successfully used to identify the in-flight aircraft transfer function relating aircraft motion to pilot stick movement, in which the PNS stimulus was generated by the pilot responding to a flashing light display (33).

In order to check the influence of measurement and extraneous noise on spectral analysis estimates, the coherency function,

$$\gamma^2 = \frac{|G_{zx}|^2}{G_{xx} G_{zz}} = \frac{G_{yy}}{G_{yy} + G_{nn}} \quad [40]$$

is used, values of  $\gamma^2 = 1$  resulting from tests on a completely noise free linear system in which case  $G_{yx}(\omega) = G_{xy}(\omega)$ , which is the quantity actually observed. The signal-to-noise ratio for the SUT is then related to the coherency function by the expression

$$\left[ \frac{S}{N} \right] = \frac{\gamma^2}{1 - \gamma^2}$$

which may be used at the SUT development stage as a guide to choosing a suitable test signal spectrum.

## 12. 'Closed Loop' Testing

All results so far discussed have been obtained under open-loop conditions, in which the aircraft loop is not closed. The present author has participated in ground based 'closed loop' tests in which the aircraft aerodynamics and kinematics were analogue simulated, the actual aircraft autopilot and actuators being stimulated during the test, as shown in Fig.15. The aircraft was situated adjacent to the simulator which must be calibrated against flight trials if a high level of confidence is to be achieved using this method. Fig.15 also shows the autoland phase simulated during the experimental work, and indicates the three heights chosen for detailed small perturbation tests using PNS and cross-correlation. The aircraft response varies considerably during the autoland phase thus changing  $\phi_{xy}(\tau)$  as a function of height, and it was confirmed that the effect of changes in autopilot parameters is also observable in  $\phi_{xy}(\tau)$ . (34)

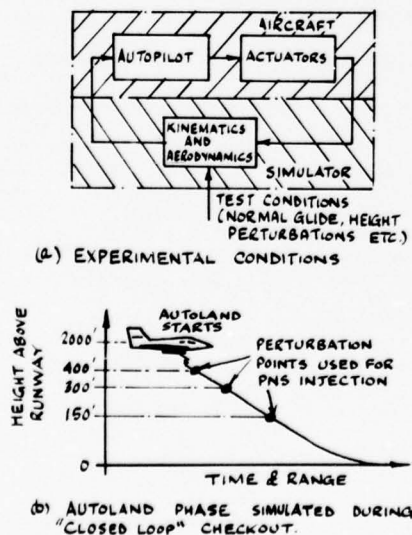


FIG.15. "CLOSED LOOP" TESTING OF AIRCRAFT AUTOLAND SYSTEM.

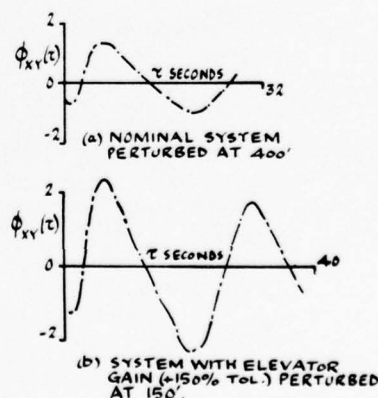


FIG.16. SAMPLE CORRELATION FUNCTIONS FOR AUTOLAND SYSTEM.

Two sample results are shown in Fig.16, the initial negative excursion being a well known result for an SUT with a system zero in the right hand plane. At the time the experiments were conducted, the simulator exhibited reliability problems which would no doubt be overcome by present day technology. It was not reasonably proven for this particular aircraft that the addition of the simulator increases confidence in the operational status of the aircraft, mainly because the sensors are not excited during the test so that



sub-system testing was implemented during maintenance. However in other areas, 'closed-loop' testing via simulation is being increasingly used, particularly in the automatic testing of weapon systems, so that the method still warrants consideration for specific applications.

### 13. Conclusions

Dynamic testing is now a universal method of assessing the operational status of a wide variety of systems ranging from amplifiers at one end of the spectrum to complete aircraft autoland systems at the other. The advent of the FFT algorithm, coupled with the ready availability of digital computers has had a considerable effect on the implementation of dynamic test techniques. It cannot be emphasised too strongly that it is the dynamic test data itself which is fundamental and the method of obtaining it is secondary to the objective of selecting those test features which adequately discriminate between 'sick' and 'healthy' systems. It is far better to undertake a manually controlled dynamic test of very simple form rather than to have no dynamic test at all. It is hoped that this paper has adequately reviewed the basic guide lines to be adopted in dynamic test stimulus and measurement feature selection.

### 14. Acknowledgements

Much of the work described in this paper has been supported by the Science Research Council, the Ministry of Defence (Procurement Executive) and U.K. Industry. This support is gratefully acknowledged as is the opportunity to participate in such a highly stimulating activity.

### References

1. J. Baumeister. 'Inflight Analysis of Aerospace Vehicle Dynamics by Correlation Techniques', NASA Computation Laboratory Memorandum, Dec. 1972.
2. V.S. Levadi. 'Pattern Recognition Applied to Fault Detection', Proc. IEEE Convention on Military Electronics, Washington, D.C., pp 197-202 Sept. 1965.
3. F.R. Gill. 'The Integrity of a Civil Blind Landing System with Particular Reference to the Azimuth Channel' RAE Tech. Report No. 65022, Feb. 1965.
4. A.H. Parker and A.H. Stewart. 'Automatic Test Systems for Production and Maintenance', Proc. IERE Symposium on ATE, Cambridge, U.K. pp 283-311 July, 1968.
5. H. Sriyananda and D.R. Towill. 'Fault Diagnosis Using Time Domain Measurements', Radio and Electronic Engineer, 43, 9, pp 523-533, 1973.
6. H. Sriyananda, D.R. Towill and J.H. Williams. 'Voting Techniques for Fault Diagnosis from Frequency Domain Test Data', IEEE Trans. on Reliability, R-24, No.4, pp 260-267, 1975.
7. R.F. Garcia. 'Fault Isolation Computer Methods' NASA Contractor Report CR-1758, 1971.
8. W.F. Fielding (Transl). 'Flight Control System for the Concorde Supersonic Civil Transport Aircraft' RAE Lib. Trans. 1615, 1972.
9. R.J. Allen. 'Failure Prediction Employing Continuous Monitoring Techniques' IEEE Trans. ASI, No.2, pp 924-930, 1963.
10. D.R. Towill. 'Transfer Function Techniques for Control Engineers', Iliffe Book Co., London, 1970, chapters 4 and 5.
11. D.R. Towill and J.D. Lamb. 'Pseudo-Random Signals Test Non-Linear Controls', Control Engineering, 17, No.11, pp59-63, 1970.
12. G.J. Thaler and R.G. Brown. 'Analysis and Design of Feedback Control Systems' McGraw-Hill Book Co. Ltd. New York, Chapter 8, 1960.
13. D.R. Towill, J.S. Cooper and J.D. Lamb. 'Dynamic Analysis of Fourth Order Feedback Control Systems', Int. Jnl. Control, 10,2 1969, pp 121-146.
14. J.M. Brown, D.R. Towill and P.A. Payne. 'Predicting Servomechanisms Dynamic Errors from Frequency Response Measurements', Radio and Electronic Engineer, 42, 1, pp 7-20, 1972.
15. D.R. Towill and Z. Mendi. 'Prediction of the Transient Response of High Order Linear Systems Using Low Order Models', Trans. IMC, 3,1, pp T1-T9, 1970.
16. W.H. Reed III, A.W. Hall and L.F. Barker, Jr. 'Analogue Techniques for Measuring the Frequency Response of Linear Physical Systems Excited by Frequency Sweep Inputs', NASA Report No. IND-508, 1960.
17. J.E. Gibson, Z.V. Rekasius, E.S. McVey, S. Sridhar and C.D. Leedham. 'A Set of Standard Specifications for Linear Automatic Control Systems', Trans AIEE, Applications and Industry, 80, pp 65-77, 1961.
18. D.R. Towill. 'Dynamic Test Techniques Utilised in ATE' REME Journal No. 22 pp 35-43, 1972.
19. S. Lees. 'Interpreting Dynamic Measurements of Physical Systems', Trans ASME, 80, pp 833-57, 1958.
20. C.S. Elsdon and A.J. Ley. 'A Digital Transfer Function Analyser Based on Pulse Rate Techniques', Automatica, 5, No.1, pp 51-60, 1969.
21. J.D. Lamb and P.A. Payne. 'Noise Rejection Properties of Modern Measurement Techniques in Control Engineering', Proc. 5th Asilomar Conf. on Circuits and Systems, pp 548-554, 1971.
22. D.R. Towill and P.A. Payne. 'Frequency Domain Approach to Automatic Testing of Control Systems', Radio and Electronic Engineer, 41 pp51-60, 1971.
23. M.T.G. Hughes and A.R.M. Noton. 'Measurement of Control System Characteristics by Means of a Cross-Correlator', Proc IEE, 109, Pt B, No.43, pp77-83, 1962.
24. D. Graupe. 'Identification of Systems', Van Nostrand Reinhold Company, New York, 1972, Chapter 4.
25. W.D.T. Davies. 'Random Signal Testing for Evaluating System Dynamics', Wireless World, pp 407-412, Aug. 1966.
26. J.D. Lamb. 'Use of Pseudo-Random Binary Signals for the Production Testing of Dynamic Systems', Ph.D. Thesis, UWIST, 1970.
27. P.A. Payne. 's Plane Techniques for the Production Testing of Feedback Control Systems', Ph.D. Thesis, UWIST, 1972.
28. G.E. Driefke, J.O. Hougen and G. Mesmer. 'Effects of Truncation on Time to Frequency Domain Conversion' ISA Transactions, pp 353-368, Oct. 1962.
29. J.P. McCarthy and P.W. LaClair. 'Transfer Function Testing', IEEE ASSC Record, pp 1-5, 1972.
30. D.E. Newland. 'An Introduction to Spectral Analysis' Longmans, London, 1975, Chapter 10.
31. D.A. McCallister and L.L. Bickford. 'Frequency Response Determination Using White Noise Excitation' Journal Spacecraft and Rockets, Vol. 5, No. 7 pp 873-874, July 1968.
32. C.W. Skingle. 'A Method for Analysing the Response of a Resonant System to a Rapid Sweep Input', RAE Tech. Report 66379, Dec. 1966.



33. D.E. Fry 'Use of Cross-Correlation and Power Spectral Techniques for the Identification of Hunter Mark II Dynamic Response' RAE Tech. Report No.69156, July 1969.
34. J.D. Lamb, A.R. Pankhurst and D.R. Towill. 'Correlation Techniques Applied to the Dynamic Testing of Aircraft Autoland Systems', Proc. 7th Int. Aerospace Symposium, Cranfield, pp 23.1-23.7, 1972.

### Nomenclature

#### Abbreviations

ATFCS	Automatic terrain following system
SUT	System under test
ATE	Automatic test equipment
FFT	Fast Fourier Transform
PNS	Pseudo noise sequence
[S/N]	Signal to noise ratio

#### Principal Symbols

s	Laplace operator
H(s)	transfer function of SUT
$a_i, b_i$	coefficients of $s^i$ in system transfer function numerator and denominator respectively
n, q	order of polynomials in s
X	SUT input signal
Y	SUT output signal
h(t)	SUT impulse response
u(t)	SUT step response
r(t)	SUT ramp response
K	SUT d.c. gain
$T_i, T_j$	SUT time constants
$\zeta_i, \zeta_j$	SUT damping ratios
$\omega_{ni}, \omega_{nj}$	SUT undamped natural frequencies
$\prod$	product sign
$z_i$	SUT zeros
$p_j$	SUT poles
$A_j, \phi_j$	residue terms in SUT transient response
$\lambda$	ratio of undamped natural frequencies
t	time
T	transpose (in test feature analysis)
T	correlation time
$F_i$	i <sup>th</sup> feature used to checkout SUT
$\pm g$	checkout gate width set on i <sup>th</sup> test feature
$\alpha_j$	j <sup>th</sup> parameter affecting performance of SUT
$\omega, f$	excitation frequency
$\phi$	SUT phase lag
$ H(j\omega) $	SUT amplitude ratio
$\phi$	correlation function
n(t)	measurement noise
$\tau$	time delay used in correlation function
a	peak amplitude of PNS pulse and of sinusoidal stimulus
n	harmonic number
N	$\omega T/2\pi$ (in sinusoidal testing)
N	$(2^R - 1)$ (in PNS testing)
$S_p$	peak step response of SUT
$I_p$	peak impulse response of SUT
R	number of stages in PNS shift register
$\Delta t$	PNS clock period
$\sigma$	standard deviation
$\rho$	spectral density of white noise
r	number of sequences of PNS over which correlation takes place
$S(\omega)$	spectral density of signal
$A_i$	$R_i + jQ_i$

**PART III**

**DESIGN AND IMPLEMENTATION**

# THRESHOLDLESS REDUNDANCY MANAGEMENT WITH ARRAYS OF SKEWED INSTRUMENTS

by  
J.E. Potter  
and  
M.C. Suman

Northrop Corporation, Electronics Division  
Precision Products Department  
100 Morse Street, Norwood, MA 02062

## INTRODUCTION

When redundant arrays of skewed instruments are considered as a means of achieving higher reliability, it is assumed that the system is able to detect and isolate the presence of a "failed" instrument. However, it is difficult to establish the resulting system performance if the failure detection algorithm uses explicit thresholds to reject failed instruments. Details about the distribution of errors in real instruments are seldom known with sufficient accuracy to justify the underlying statistical assumptions used in setting such thresholds. Thresholdless detection algorithms, which do not depend on detailed knowledge of instruments error statistics, are therefore of special practical significance.

As an example of a thresholdless detection algorithm, consider the so-called mid-value selection technique. When three instruments are used to measure the same scalar quantity, such as one component of vehicle angular rate, a common technique is to take the middle value, or median, as the estimate of the measured quantity. Advantages of this technique are that it employs simple logic; requires no statistical information about the instruments for its implementation; tolerates one instrument failure with relatively little degradation in system performance; and operating with three unfailed instruments, gives errors smaller, on the average, than when only a single instrument is used.

Applying the mid-value selection technique directly to the measurement of a 3-dimensional vector quantity requires a configuration of nine instruments, with three instruments aligned along each input axis. Nearly equivalent redundancy can be achieved with five or six skewed instruments, but in these configurations there is no "middle value" to select. However, by reformulating mid-value selection as an algorithm which minimizes an appropriate performance index, it is possible to extend its application to arbitrarily skewed arrays. This extension then yields bounds on the system performance which can be achieved in real worst-case situations.

## REDUNDANT SENSOR SYSTEMS

Redundant systems are considered for several different reasons. One of these is simply to provide a backup so that an operating system may be "switched out" for routine maintenance or repair; this practice is common among power generating systems, and is found in commercial aircraft navigation when an extra inertial measurement unit is carried aboard in a non-operational spares rack. Of more importance for the purpose of improving aircraft flight integrity are uses of redundancy which improve operational reliability (in the sense of increasing the probability of mission success) by allowing the system to withstand failures when immediate repairs cannot be made. The significant distinction between these two classes of redundant systems is that the latter must have the ability to detect and isolate their own failures, and must be capable of meeting their operational performance requirements even after such failures have occurred. As will be seen, it is this latter consideration which most strongly influences system design.

It should be emphasized that great improvement in system reliability is gained by the use of even a modest degree of sensor redundancy. For example, the probability of failure of a three-dimensional measuring system with three single-axis sensors is about three times the failure probability of a single sensor (since the probabilities are much less than one). For a five sensor system which will still meet its requirements after one instrument has failed, the probability of system (mission) failure is reduced to about 10 times the square of the single instrument failure probability. Thus for an instrument failure probability of  $10^{-3}$ , the system failure probability is only about  $10^{-5}$ , and this probability will be reduced by an order of magnitude for each additional redundant instrument.

One reason often advanced for considering redundant sensors is that improved system performance can be achieved because of the averaging effect of having several independent measurements. Unfortunately, substantial improvement occurs only when a very large number of instruments is used. For example, when a vector quantity (e.g., acceleration or angular rate) is measured with three orthogonal single-axis instruments, the error in estimating each vector component is simply the error of the respective instrument. When four such instruments are used in an optimal geometric configuration, the error in estimating each component is reduced to about 80% of the instrument errors. To cut the estimation error in half requires 12 instruments in an optimal array, and to reduce it by an order of magnitude would require about 300 instruments.

Thus, as a practical matter, the use of redundant sensors to obtain improved system performance is seldom economically justifiable. In fact, as will be shown below, where redundancy is used to provide complete failure tolerance, the over-all system performance will necessarily be less than that which would be specified for a non-redundant system where such failure tolerance is not required.

## SYSTEM CONFIGURATIONS

We assume for simplicity that all systems under consideration consist of single-degree-of-freedom measuring instruments. That is, each instrument measures the projection of the input vector along a single predetermined instrument "input" axis. In practice, such instruments will be accelerometers, gyros, or closely related sensors, together with their functionally inseparable electronics (such as torque rebalance servos).

There is no essential loss in generality caused by ignoring more complex sensors, like two-axis accelerometers and two-degree-of-freedom gyros. The analysis of redundant systems composed of such instruments is similar provided that 1) allowance is made for correlation between the measurement errors found in separate axes of the same instrument, and 2) consideration is given to "ambiguous" failure modes whereby detection of a failure on one measurement axis may or may not signal a failure of the instrument as a whole.

Many different configurations (geometric arrangements) of sensors have been proposed and analyzed in the literature.<sup>(1), (2), (3), (4), (5)</sup> However, apart from the total number of instruments used and the requirement that instrument axes be "reasonably" distributed, the actual configuration has little effect upon the redundancy management approach.

The imprecise phrase "reasonably" distributed is needed to avoid a plethora of special considerations and qualifications. In order to measure a three-dimensional quantity, at least three distinct measurement axes are required, not all lying in the same plane. It is not necessary that they be orthogonal, but as a practical matter it is desirable that they be roughly so, in order not to distort badly inputs lying in special directions (widely skewed arrays have been used to desensitize the instruments to certain directions). When redundant instruments are used, it is normally desirable that no resulting set of three should be coplanar so that each instrument can provide redundancy to all the others; to minimize distortion, each set of three should approach as far as possible an orthogonal array. On the other hand, an often-used configuration has multiple instruments colinearly arranged along the three orthogonal directions. Such an array is far from efficient, but it can be included in the general theory by adding sufficient qualifications. However, for simplicity here, we always assume for the redundant arrays under consideration, that all sets of three instruments are non-coplanar.

Most efficient arrangements fall in one of two categories: those with an odd number of instruments uniformly distributed around a cone, and those with an even number of instruments, one of which lies along the central axis of the cone. The cone angle may be selected to equalize the estimation error in all directions: for  $n$  instruments with equal variances, this angle for the first category is found to be 54.74 degrees (the same as for three orthogonal instruments); for the second category, it is found to be that angle the square of whose cosine is  $(n-3)/(3n-3)$ .<sup>(3)</sup>

Other arrangements will have somewhat poorer over-all performance. In practice, other engineering considerations usually determine which configuration should be used. For example, with five instruments arranged on the 54.74 degree cone, the error in estimating a component of the input from all five measurements will be about  $0.77\sigma$ , or about  $0.95\sigma$  if only four of the measurements are used. When the cone angle is opened up to 63.43 degrees (the dodecahedron), these errors will be increased to about  $0.82\sigma$  and  $1.0\sigma$ , respectively. However, this latter arrangement turns out to allow a much simpler physical mounting structure, and this consideration will probably lead to its choice in practice.

Certain configurations are especially convenient for analytical purposes. Among the more popular of these is the six instrument array with the input axes normal to the faces of a regular dodecahedron. Because of the extent to which this configuration has been treated in the literature, a number of misconceptions are often encountered when discussing alternatives. One such misconception is the belief that it is the special geometry of the dodecahedron which yields well-defined parity equations and simple algorithms for redundancy management. In fact, all configurations have parity equations, and except in very special circumstances, impose approximately the same computational complexity.

If  $\underline{h}$  is a unit vector in the direction of the input axis of a given instrument, then the output  $m$  in response to a (vector) input  $\underline{x}$  will be

$$m = \underline{h}^T \underline{x} + \epsilon, \quad \text{with } \underline{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \quad \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

where  $\epsilon$  is the error in the measurement. For convenience, the constant of proportionality (scale factor) is neglected.



When there are  $n$  instruments in the system, there will be  $n$  such equations, so we may write

$$\underline{m} = H \underline{x} + \underline{\epsilon}$$

with  $\underline{m} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix}$ ,  $\underline{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ \vdots & \vdots & \vdots \\ h_{n1} & h_{n2} & h_{n3} \end{bmatrix}$$

The matrix  $H$  is called the measurement matrix, and the rows of  $H$  will be referred to as the measurement axes.

We assume without further comment throughout the following that the rank of  $H$  is equal to the dimension (normally 3) of the measured vector. It cannot be more, and if it were less there would be a component of the measured vector which would not affect the values of the measurements. Those configurations in which this might occur are excluded by the "reasonableness" criterion discussed above.

As an example, for a "pentad" array formed from five instruments with input axes arranged normal to five of the six faces of a regular dodecahedron, we can write (depending on the labeling of the instruments)

$$H = \begin{bmatrix} s & 0 & c \\ -s & 0 & c \\ c & s & 0 \\ c & -s & 0 \\ 0 & c & s \end{bmatrix}$$

$$c = \cos \alpha = \sqrt{\frac{5 + \sqrt{5}}{10}} = 0.85065081$$

$$s = \sin \alpha = \sqrt{\frac{5 - \sqrt{5}}{10}} = 0.5257311$$

$$\alpha = \frac{1}{2} \tan^{-1} 2 = 31.7175^\circ$$

We will refer to this array in the examples below, but it should be noted that there is no particular significance to this choice, and that it is made simply for analytical convenience.

## MEASUREMENT ERRORS

All inertial sensors have various kinds of deterministic and non-deterministic errors. These include constant bias offsets, scale factor errors, and sensitivity to spurious input and environmental factors. It is assumed throughout this paper that all instruments under consideration have been calibrated and compensated to the extent that they may be adequately described as having unit scale factors and random additive errors with zero mean and constant variance  $\sigma^2$ . When multiple instruments are considered, they are all assumed to have the same variance. The (important) case of scale factor errors has been ignored to simplify the discussion.

Except for the assumption that errors in separate instruments are statistically independent, no special assumptions are made about the distribution of the instrument errors. It is the authors' belief that seldom (if ever) in practice is sufficient data on the distribution of real instrument errors available to justify the use of algorithms based on higher order statistics. This is especially true in those high reliability cases where redundancy management techniques are finding most application. The instruments which are selected are those which have been shown to have had very few failures, so that there is little in the way of failure data on which to base statistics.

## FAILURES, FAILURE DETECTION, AND REDUNDANCY MANAGEMENT

The question of what constitutes a "failed" instrument involves certain (phenomenological) subtleties. These are reflected in the fact that situations exist, such as when the quantity being measured is zero, in which the behavior of a perfectly functioning instrument is indistinguishable from the behavior of one which is not working at all. In practice instruments often fail by small degrees so that the indicated measurement becomes contaminated with nearly unobservable error. For operational purposes we can define an instrument as failed when it is contributing measurement errors sufficiently large as to jeopardize the system mission.

In an engineering sense an instrument is "good" only if it is somehow capable of measuring with the required degree of accuracy all of those inputs to which it might be subjected. The attempt to measure this capability in practice results in the inclusion of built-in test equipment (BITE), and the use of special detectors and circuitry to continuously monitor physical parameters such as supply voltage, case temperature, wheel sync, and torque resistance. There are two serious problems with this "hardware" approach, apart from the fact that the available set of useful parameters is quite limited. The first is that an out-of-tolerance condition on one of the monitored

parameters is at best only an implication of possible instrument failure. It is quite frequently found that an instrument which is not working correctly is working well enough for the mission to succeed. The second problem is that all additional monitoring circuitry brings with it the possibility of additional failure modes, and a new requirement to monitor the failures in the failure monitors. The analysis of such systems often leads to a condition which has been termed sylogistic instability: "this circuit is good unless this voltage drops, which can only happen if this other circuit is bad, unless...". The chain of logic involves a kind of feedback, and it may be very difficult or impossible to be sure the system will work as intended under all conditions. It is perfectly possible to design a "fail-safe" system which is permanently disabled by its own logic.

Redundant systems provide a way out of this difficulty by allowing decisions regarding failed instruments to be based on the outputs of the instruments themselves: all other (possibly erroneous) failure indications can therefore be ignored, at least until the level of redundancy has been reduced by realized failures to where the other information must be used as a "last resort". This basic approach to failure detection has been called the "algorithmic covering technique". Clearly what it imposes on the redundancy management algorithm is the requirement that it be able to isolate or tolerate any failure large enough to jeopardize the mission.

To isolate or tolerate a single failure requires that the number of instruments must exceed the dimension of the measured quantity by at least two. Thus five instruments are required to tolerate a single failure in 3-dimensional space. If the minimal number of instruments required to tolerate one failure is used, the reliability of the system is just the probability that two instruments don't fail. If more than the minimal number of instruments is used, then the system may be able to tolerate additional failures. However, a difficulty is introduced by the need to consider the possibility of simultaneous failures (simultaneous in the sense that one occurs before another has been completely isolated). To see this, consider the case of six instruments in 3-dimensional space. When a single failure occurs and can be isolated, the remaining configuration of five instruments possesses sufficient inherent redundancy to isolate a second failure. But if two failures occurred simultaneously, we might not be able to say which set of four was still good unless we knew which sets of five contained at most one failure. In fact it has been shown<sup>(6)</sup> that to be sure of isolating  $k$  failures among  $L$  instruments in  $n$ -dimensional space, we must have  $L \geq 2k + n$ . Thus to provide certain failure tolerance of two failures under worst-case conditions in 3-dimensional space requires the use of seven sensors. To avoid the complications inherent in multiple failure tolerance, we will discuss below only systems designed to tolerate a single failure.

In accordance with the algorithmic covering technique, failure detection must depend only on the disagreement or "inconsistency" between redundant measurements. As will be shown, all information about this disagreement resides in a set of linear relationships called parity equations. If the number of independent parity equations were equal to the number of measurements, then the equations could be solved to yield the measurement errors. The fundamental difficulty in identifying failed instruments arises from the fact that there are necessarily fewer independent parity equations than instruments. This limitation will be explored in what follows.

A number of interesting redundancy management algorithms have been proposed and analyzed in the open literature.<sup>(4), (7), (8), (9), (10), (11), (12)</sup> Some of these may offer significantly better estimates under some conditions.<sup>(13)</sup> A real difficulty is that the "best estimate" under some conditions is not usually the "best strategy" under most conditions. An example will help to clarify this distinction.

Suppose that the same component of angular rate is measured (in  $^\circ/\text{hr.}$ , for example) with three different gyros, each having the same standard error of 0.1, and giving 1.4, 1.2, and 1.9 respectively. If we think that all three gyros are good, then our best estimate of the rate is probably 1.5, their arithmetic mean. The standard error of this estimate is about 0.06. On the other hand, if we think that the third instrument is so far out that its value ought to be disregarded, then our best estimate is probably 1.3, the arithmetic mean of the other two. The standard error of this estimate is about 0.07.

Unfortunately, if we assume the first case when the second is true, we shall be off from the best estimate by 0.2, and if we assume the second case when the first is true we shall also be off by 0.2. We will be somewhat safer if we choose the middle value 1.4, since we will then be off by 0.1 at most, regardless of which case is actually true. However, the standard error in this estimate (the median of 3) is about 0.08, which is somewhat larger than either of the other two. Restated, we choose the mid-value not because it is the best estimate — it is in fact known not to be so good — but rather because it differs from the true value by the least amount, in the worst case, when a (single) instrument has failed. It is the precise restatement of the mid-value selection technique in this form which we will generalize below.

# PARITY EQUATIONS

If a pair of measurements  $m_1, m_2$  are made of the same physical quantity then, in the absence of error

$$m_1 - m_2 = 0$$

Such an equation is called a Parity Equation, since it asserts the equality of the two measurements. When the measurements contain errors the parity equation becomes

$$(m_1 - \epsilon_1) - (m_2 - \epsilon_2) = 0$$

or, equivalently

$$m_1 - m_2 = \epsilon_1 - \epsilon_2 = \eta$$

The value  $\eta$  may be considered a measure of the lack of agreement or "inconsistency" in the measurements. Note that it is not directly a measure of the error, since there are many combinations of measurement errors for which  $\eta = 0$ . When a third measurement  $m_3$  is considered, there are three possible parity equations which can be formed

$$\begin{aligned} m_1 - m_2 &= \epsilon_1 - \epsilon_2 = \eta_1 \\ m_2 - m_3 &= \epsilon_2 - \epsilon_3 = \eta_2 \\ m_3 - m_1 &= \epsilon_3 - \epsilon_1 = \eta_3 \end{aligned}$$

It is important to notice that the third equation is a linear combination of the other two; in a strict sense there are only two degrees of inconsistency between the three measurements. Consideration of a fourth measurement will introduce three more parity equations, but only one more degree of inconsistency.

When a 3-dimensional vector quantity  $\underline{x}$  is measured with a redundant array of instruments, the component of  $\underline{x}$  along any one instrument axis can be written in terms of its components along any other three. For example, consider four measurements,  $m_1, m_2, m_3$  and  $m_4$ . We can write

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = Q \underline{x}, \quad \text{with } Q = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Then

$$m_4 = h_4^T \underline{x} = h_4^T Q^{-1} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$$

Thus, there is an equation of the form

$$\lambda_1 m_1 + \lambda_2 m_2 + \lambda_3 m_3 + \lambda_4 m_4 = 0$$

which is uniquely determined (up to a constant multiplier) by the geometric orientation of the instruments. When measurement errors are included we will have

$$\lambda_1 (m_1 - \epsilon_1) + \lambda_2 (m_2 - \epsilon_2) + \lambda_3 (m_3 - \epsilon_3) + \lambda_4 (m_4 - \epsilon_4) = 0$$

or, equivalently

$$\lambda_1 m_1 + \lambda_2 m_2 + \lambda_3 m_3 + \lambda_4 m_4 = \eta$$

We will call such an equation a parity equation by analogy with the one-dimensional case. When the measurements are without error  $\eta = 0$ . We call  $\eta$  the inconsistency<sup>(7)</sup> in the measurements which, though made along different directions are not physically independent. Note, however, that as explained earlier,  $\eta$  is not directly a measure of the error, since there are many combinations of measurement errors for which it also vanishes.

Now, if there are in fact  $\ell$  instruments ( $\ell > 3$ ), then there are  $C_4^\ell = \frac{\ell!}{4!(\ell-4)!}$  distinct combinations of four measurements which can be used to form different parity equations. Thus for five-instrument arrays, there are five distinct parity equations (that is, distinct up to an arbitrary constant multiplier), and for six-instrument arrays there are 15 distinct parity equations. It is important to note that not all of the parity equations are independent. In fact, the number of independent equations  $q$  is given by

$$q = \ell - n$$

Where  $\ell$  is the number of measurements and  $n$  is the dimension of the measured vector. Thus for five skewed instruments in 3-dimensional space there are only two independent parity equations, and for six skewed instruments only three. The consequences of this will be explored in detail below.

#### A LOWER BOUND ON THE LARGEST ERROR

Since the coefficients  $\lambda$  in each of the parity equations are determined only up to an arbitrary multiplier, we may assume without loss of generality that they have been normalized so that the sum of the absolute values of the coefficients in each equation is unity. The significance of this particular normalization is that we can then say of the measurements in any particular parity equation that there is at least one error whose magnitude is as large as the magnitude of the "inconsistency"  $\eta$ . This fact is the basis of the Minimax failure detection method of Potter and Deckert.<sup>(11)</sup>

#### PARITY SPACE

Since there are only  $q = \ell - n$  independent relations among the  $\ell$  parity equations, it is possible to express the inconsistency values entirely in terms of  $q$  independent variables, say  $p_1, p_2, \dots, p_q$ . The vector  $\underline{p}$  will be called a parity vector, and the  $q$  dimensional space of all parity vectors will be called parity space. All information contained in the  $C_n^\ell$  parity equations is available from the  $q$  components of the parity vector. Because the dimension of the parity vector is less than the dimension of the measurement vector  $\underline{m}$ , considerable simplification and insight is provided by expressing failure detection algorithms in terms of the parity vector.

Now, the parity equations are special linear combinations of the measurements whose values depend only on the measurement errors. These values are independent of the value of the measured vector. As defined above, each parity equation involves only four measurements, but it is clear that we could have defined each equation as a linear combination of all components of the measurement vector  $\underline{m}$ , so that each equation could be written in the form

$$\eta = \underline{v}^T \underline{m}$$

where  $\underline{v}^T$  is an appropriately defined  $\ell$ -dimensional vector (some of whose components are zero).

To slightly generalize the concept behind the parity equations, let us define a parity function as any (possibly non-linear) function of the measurement vector whose value is independent of the measured vector. Thus the parity equations defined above are linear parity functions in this sense.

For an arbitrary (linear) parity function we have

$$\underline{v}^T \underline{m} = \underline{v}^T (H\underline{x} + \underline{e})$$

Since by definition this equation must be independent of  $\underline{x}$  it follows that  $\underline{v}^T H = 0$  or, equivalently  $H^T \underline{v} = 0$ . In other words, the most general (linear) parity functions are simply the inner products of the measurement vector  $\underline{m}$  with the null vectors of the matrix  $H^T$ .

To construct a parity vector  $\underline{p}$ , first let  $V$  be a  $q \times \ell$  matrix whose rows are any  $q$  linearly-independent null vectors of  $H^T$ . Because the  $\ell \times n$  matrix  $H$  has rank  $n$ , the dimension of the null space of  $H^T$  is  $q = \ell - n$  and such a set of vectors can be found. For example, the rows can be taken simply as the coefficients of the first  $q$  parity equations. Then by construction  $VH = 0$ , and we may take as a parity vector the  $q$  dimensional vector  $\underline{p} = V\underline{m}$ . In order to show that  $\underline{p}$  is a parity vector in the sense defined above, we must show that the values of all the parity functions can be derived from it.



To do so, let  $K$  be any  $n \times \ell$  left inverse of  $H$  so that  $KH = I$ , and define  $\underline{x}_b = K\underline{m}$ . Since the rank of  $H$  is  $n$  and  $\ell > n$ , an infinity of left inverses exists.

Then we can write

$$\begin{bmatrix} \underline{x}_b \\ \underline{p} \end{bmatrix} = \begin{bmatrix} K\underline{m} \\ V\underline{m} \end{bmatrix} = \begin{bmatrix} K \\ V \end{bmatrix} \underline{m}$$

and  $\underline{m}$  can be obtained by inversion provided the  $\ell \times \ell$  matrix  $A = \begin{bmatrix} K \\ V \end{bmatrix}$  is non-singular.

To prove that  $A$  is non-singular, let  $\underline{a}$  be an appropriately partitioned null vector of  $A^T$  so that

$$A^T \underline{a} = K^T \underline{a}_1 + V^T \underline{a}_2 = 0$$

Then

$$H^T A^T \underline{a} = H^T K^T \underline{a}_1 + H^T V^T \underline{a}_2 = \underline{a}_1 = 0$$

From this we conclude that  $V^T \underline{a}_2 = 0$

And, since the columns of  $V^T$  are linearly independent,  $\underline{a}_2 = 0$ .

Thus, the only solution of  $A^T \underline{a} = 0$  is  $\underline{a} = 0$ ,  $A^T$  is non-singular, and hence  $A$  is non-singular.

Now, partition  $A^{-1}$  in terms of  $\ell \times n$  and  $\ell \times q$  matrices  $B, M$  so that  $A^{-1} = [B, M]$ .

Then

$$\begin{aligned} H &= A^{-1} A H = [B, M] \begin{bmatrix} K \\ V \end{bmatrix} H \\ &= [BK + MV] H = B \end{aligned}$$

since  $KH = I$  and  $VH = 0$  by construction.

Hence

$$\underline{m} = A^{-1} \begin{bmatrix} \underline{x}_b \\ \underline{p} \end{bmatrix} = [B, M] \begin{bmatrix} \underline{x}_b \\ \underline{p} \end{bmatrix} = H\underline{x}_b + M\underline{p}$$

and any linear parity function can be written

$$\eta = \underline{v}^T \underline{m} = \underline{v}^T H\underline{x}_b + \underline{v}^T M\underline{p} = \underline{v}^T M\underline{p}$$

which was to be shown.

#### BASE VECTORS AND THE LEAST SQUARES ESTIMATE

It is convenient at this point to introduce the concept of a base vector. Consider the vector  $\underline{x}_b$  defined above. We have

$$\underline{x}_b = K\underline{m} = K[H\underline{x} + \underline{\epsilon}] = \underline{x} + K\underline{\epsilon}$$

The vector  $\underline{x}_b$  equals  $\underline{x}$  when there are no errors, and approximates it when the errors are small. We call any such vector a base vector for  $\underline{x}$ .

Note from the previous section that the measurement vector,  $\underline{m} = H\underline{x}_b + M\underline{p}$ , is composed of two components, one defined on the base vector and one on the parity vector. The first,  $\underline{m}_b = H\underline{x}_b$ , may be thought of as the "consistent" part of the measurement, since if  $\eta$  is any parity function  $\eta(\underline{m}_b) = 0$ . The second part,  $\underline{m}_r = M\underline{p}$ , will be called the reduced measurement. It may be used instead of the actual measurement vector  $\underline{m}$  in evaluating any parity functions, since  $\eta(\underline{m}_r) = \eta(\underline{m})$ .

The base vectors for  $\underline{x}$  are not uniquely determined. In particular, any solution for  $\underline{x}$  which is derived from  $n$  of the  $\ell$  measurements is a valid base vector. The significance of the base vectors is that they are all potential candidates to be used for the estimate of  $\underline{x}$  in any redundancy management algorithm.

One base vector of particular importance is the so-called least squares estimate. Where more than the minimum number of instruments is available, the "best estimate in the least squares sense" is that value which minimizes the square of the length of the error vector, that is

$$\underline{\epsilon}^T \underline{\epsilon} = (\underline{m} - H\underline{x})^T (\underline{m} - H\underline{x})$$

Let  $\hat{\underline{x}}$  be the solution to the (normal) equations

$$H^T H \hat{\underline{x}} = H^T \underline{m}$$

Then, with some manipulation, we can get

$$\underline{\epsilon}^T \underline{\epsilon} = (\underline{m} - H\underline{x})^T (\underline{m} - H\underline{x}) + (\underline{x} - \hat{\underline{x}})^T H^T H (\underline{x} - \hat{\underline{x}})$$

Since the second term on the right is a positive definite quadratic form, it follows that the minimum is obtained when  $\underline{x} = \hat{\underline{x}} = K\underline{m}$ ,

where  $K = (H^T H)^{-1} H^T$ . Note that  $\hat{\underline{x}}$  is a legitimate base vector since  $KH = I$ .

As an example, consider the pentad configuration defined above. By straightforward computation we obtain

$$\hat{\underline{x}} = (H^T H)^{-1} H^T \underline{m} = \frac{1}{2} \begin{bmatrix} s & -s & c & c & 0 \\ -c^2 s & -c^2 s & s(1+c^2) & -s(1+c^2) & 2c^3 \\ c(1+s^2) & c(1+s^2) & -cs^2 & cs^2 & 2s^3 \end{bmatrix} \underline{m}$$

Each component of the least squares estimate is simply a fixed weighted sum of the instrument outputs. Whether this estimate is used in practice depends on whether such computation is considered practical. Note, however, that in the strapdown context a computation of this form is usually used to correct for scale factor and alignment errors anyway, so that use of the estimate in practice often entails very little additional computation.

#### ORTHOGONAL PARITY SPACE

When the least squares estimate  $\hat{\underline{x}}$  is used as a base vector, the measurement equation becomes

$$\underline{m} = \hat{\underline{m}} + \hat{\underline{\epsilon}}$$

where by definition,  $\hat{\underline{m}} = H\hat{\underline{x}}$  and  $\hat{\underline{\epsilon}} = \underline{m} - \hat{\underline{m}}$

Note that  $\hat{\underline{\epsilon}}$  is really the reduced measurement  $\hat{\underline{m}}$ , so that  $\hat{\underline{\epsilon}} = M\underline{p}$ . Since  $\hat{\underline{x}}$  is by definition the value of  $\underline{x}$  which minimizes the length of  $\underline{m} - H\underline{x}$ , it is evident by simple geometry that  $\hat{\underline{m}}$  and  $\hat{\underline{\epsilon}}$  are orthogonal. Algebraically,

$$\begin{aligned} \hat{\underline{m}}^T \hat{\underline{\epsilon}} &= \underline{m}^T K^T H^T (\underline{m} - HK\underline{m}) \\ &= \underline{m}^T [K^T H^T (I - HK)] \underline{m} = 0 \end{aligned}$$

since

$$K = (H^T H)^{-1} H^T$$

and

$$K^T H^T H K = H(H^T H)^{-1} H^T H (H^T H)^{-1} H = K^T H^T = HK.$$

Although the reduced measurement is uniquely determined by the selection of a particular base vector, the parity vector is not, since we have required so far only that  $V$  be of rank  $q$  and that  $VH = 0$ . We will now derive sufficient restrictions on  $V$  in order to make the parity vector unique, and we will do this in a natural way which makes it easy to determine  $M$ . We note in passing that if  $\underline{p} = V\underline{m}$  and  $\underline{p}' = V'\underline{m}$  are any two parity vectors, then

$$\underline{p}' = V'[H\underline{x} + M\underline{p}] = V'M\underline{p}$$

From the definition of  $M$  above,

$$MV = I - HK$$

$$MVV^T = (I - HK)V^T$$

so, providing  $VV^T$  is non-singular, we have in general

$$M = (I - HK)V^T (VV^T)^{-1}$$

To prove that  $VV^T$  is non-singular, we note that  $V^T \underline{x} = 0$  and  $VV^T \underline{x} = 0$  define the same solutions:  $V^T \underline{x} = V^T$ . It would be  $VV^T \underline{x} = 0$ ;  $VV^T \underline{x}$  implies that  $\underline{x}^T VV^T \underline{x} = (V^T \underline{x})^T V^T \underline{x} = 0$  and thus, since  $V^T$  is real, that  $V^T \underline{x} = 0$ . Therefore  $V^T$  and  $VV^T$  have the same rank  $q$ , but  $VV^T$  is  $q \times q$  and thus is non-singular.

When the least squares estimate is used for the base vector, then  $(I - HK)V^T = V^T - HKV^T = V^T - [VHK]^T = 0$ . It would be convenient if we could also choose  $V$  so that  $VV^T = I$ , since then we would have  $M = V^T$ . But this can always be done: all we have required of  $V$  is that its rows be  $q$  linearly-independent null vectors of  $H^T$ . If we choose these vectors to be *orthonormal* then  $VV^T = I$  directly. Unfortunately,  $V$  is still not completely determined, since if  $U = RV$  and  $R^T R = I$ , then  $U^T U = V^T V$ . We may complete the specification of  $V$  by requiring that it be upper triangular with positive (non-null) diagonal elements. It can then be verified (we omit the proof) that if  $W = I - HK$ , then  $V$  is given by the following formulas

$$V_{11}^2 = W_{11}, V_{ij} = 0 \text{ for } j < i, V_{ij} = W_{ij}/V_{11} \text{ for } j = 2, \dots, \ell$$

$$V_{ii}^2 = W_{ii} - \sum_{k=1}^{i-1} V_{ki}^2 \text{ for } i = 2, \dots, q$$

$$V_{ij} = (W_{ij} - \sum_{k=1}^{i-1} V_{ki} V_{kj})/V_{ii} \text{ for } i = 2, \dots, q, j = i+1, \dots, \ell$$

For example, when this process is applied to the pentad described above, we get:

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 2cs & -c^2 & -s^2 & -c^2 & -s^2 \\ 0 & s & c & -s & -c \end{bmatrix}$$

To summarize, the above development defines what we will call orthogonal parity space. In orthogonal parity space we have

$$\underline{m} = \hat{\underline{m}} + \hat{\underline{\epsilon}}$$

$$\hat{\underline{m}} = HK\underline{m}$$

$$\hat{\underline{\epsilon}} = V^T \hat{\underline{p}}$$

$$\hat{\underline{p}} = V\underline{m}$$

where  $K = (H^T H)^{-1} H^T$

and where  $V$  is upper triangular, has positive diagonal elements, and satisfies

$$V^T V = I - HK$$

Therefore, it follows that

$$VV^T = KH = I \text{ and } VH = KV^T = 0.$$

Orthogonal parity space is important because of its close relationship to the concept of the "best estimate". It is particularly convenient for analytical purposes since physical symmetry in the orientation of the instruments axis tends to be preserved in geometric symmetries of the parity space.

## MEASUREMENT AXES IN PARITY SPACE

Corresponding to each of the measurement axes in real space (the physical instrument input axes), there is a special direction in parity space. If an error occurred in only one of the measurements, then the parity vector would lie along the corresponding special direction and its length would be proportional to the magnitude of the error. We will call these directions the measurement axes in parity space. Since  $\underline{p} = \underline{V}\underline{m} = \underline{V}\underline{\epsilon}$ , it is clear that they are simply the directions of the corresponding column vectors of  $\underline{V}$ .

When orthogonal parity space is used we have for the reduced measurement,  $\underline{m}_r = \hat{\underline{\epsilon}} = \underline{V}^T \underline{p}$ , so that the reduced measurement corresponding to an error in a single measurement axis is simply the projection of the parity vector on the corresponding measurement axis in parity space. However, since  $\underline{V}^T \underline{V}$  is not unity, the reduced measurement  $\underline{m}_r = \hat{\underline{\epsilon}} = \underline{V}^T \underline{V} \underline{\epsilon}$  corresponding to a single measurement error will have other non-zero components: the single error will be "distributed", and some of it will be "absorbed" into the input estimate represented by the base vector. The very need to estimate the input in the presence of error means that some of the error must contaminate the estimate. In fact, any set of errors such that  $\underline{V}\underline{\epsilon} = 0$  is necessarily unobservable, whatever estimate is chosen.

## AN EXAMPLE

In order to illustrate much of the above, we will apply the theory to the case of three measurements of a scalar (that is, a vector of dimension one) using orthogonal parity space. The example is particularly instructive because of the insight it provides into the mid-value selection technique.

Let

$$\begin{aligned} m_1 &= x + \epsilon_1 \\ m_2 &= x + \epsilon_2 \\ m_3 &= x + \epsilon_3 \end{aligned}$$

where  $m_1, m_2, m_3$  are the measurements of  $x$ , and  $\epsilon_1, \epsilon_2, \epsilon_3$  are the errors.

Then

$$\underline{m} = \underline{H}\underline{x} + \underline{\epsilon} \quad \text{with } \underline{H} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The least squares estimate for the base vector (the scalar  $x_b$ ) is

$$\hat{x} = \underline{K}\underline{m} \quad \text{with } \underline{K} = (\underline{H}^T \underline{H})^{-1} \underline{H}^T = 1/3 [1, 1, 1]$$

so that

$$\hat{x} = 1/3 (m_1 + m_2 + m_3)$$

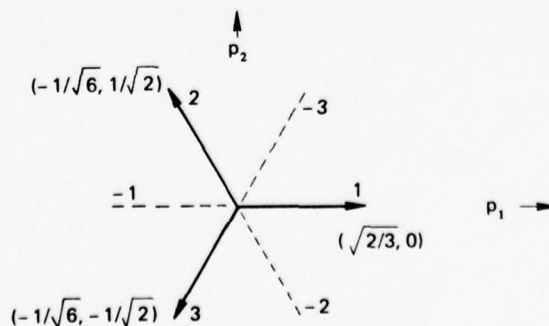
Thus the estimate (the base vector) is simply the average of the measurements, as we would expect. The matrix  $\underline{V}^T \underline{V} = \underline{I} - \underline{H}\underline{K}$  gives

$$\underline{V}^T \underline{V} = 1/3 \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Using the square root formulas from above gives

$$\underline{V} = \begin{bmatrix} \sqrt{2/3} & -1/\sqrt{6} & -1/\sqrt{6} \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

It may be verified directly that  $\underline{V}\underline{V}^T = \underline{I}$  and  $\underline{V}\underline{H} = 0$ . The columns of  $\underline{V}$  are vectors of length  $\sqrt{2/3}$  which lie along the measurement axes in parity space. Plotting these vectors gives a measurement axis every sixty degrees (one is coincident with the  $P_1$  axis).





The symmetry shown by the measurement axes in parity space reflects the symmetry of the H matrix (all measurements have equal weight).

The components of the parity vector, as given by  $\underline{p} = \underline{V}\underline{m}$ , are

$$\begin{aligned} p_1 &= (1/\sqrt{6}) (2m_1 - m_2 - m_3) \\ p_2 &= (1/\sqrt{2}) (m_2 - m_3) \end{aligned}$$

The components of the parity vector, as given by  $\underline{p} = \underline{V}\underline{e}$ , are

$$\begin{aligned} p_1 &= (1/\sqrt{6}) (2\epsilon_1 - \epsilon_2 - \epsilon_3) \\ p_2 &= (1/\sqrt{2}) (\epsilon_2 - \epsilon_3) \end{aligned}$$

We emphasize again that the parity vector is a function only of the measurement errors, and not the actual input.

The parity equations, which is to say, the linear relations between  $m_1, m_2, m_3$  which do not involve  $x$ , are simply whatever linear relations can be formed between the equations

$$\begin{aligned} 2m_1 - m_2 - m_3 &= 2\epsilon_1 - \epsilon_2 - \epsilon_3 \\ m_2 - m_3 &= \epsilon_2 - \epsilon_3 \end{aligned}$$

The most general linear parity function for this case has the form

$$\eta(\underline{m}) = am_1 + bm_2 + cm_3$$

From the requirement that  $\eta(\underline{m})$  be independent of  $x$ , we conclude that

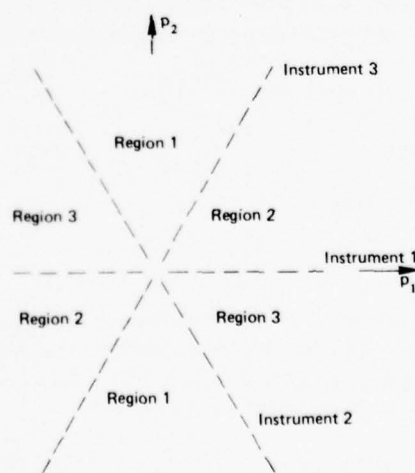
$$a + b + c = 0$$

When we express the parity function in terms of  $p_1, p_2$  we get

$$\eta(\underline{m}) = -\frac{\sqrt{3}}{2}(b+c)p_1 + \frac{1}{\sqrt{2}}(b-c)p_2$$

Thus, in terms of failure detection, we may now say: If the parity vector corresponding to a particular set of three measurements were very large (that is, relative to the magnitude of "normally expected" errors), and if it lay very nearly along one of the measurement axes in parity space, then we could presume that the instrument corresponding to that axis had failed. We see clearly from the diagram how a combination of errors on the other two axes could have caused the same result, but in that case we would have two very large failures, a "disaster" against which we are powerless to contend by any means.

If the parity vector fell somewhere in between two axes, the situation would not be so clear. We might weight the measurements if we had some (a priori) knowledge about the error statistics. But in any event, we would give special credence to the axis which is "most normal" to the parity vector, since its errors must be contributing least to the inconsistency. Or we could avoid the question of statistics entirely and use only that axis all the time. This latter is in effect the mid-value selection, and is summarized as follows.



Region	Instrument	Estimate
1	1	$m_1 = \hat{x} + (\sqrt{2/3}) p_1$
2	2	$m_2 = \hat{x} - (1/\sqrt{6}) p_1 + (1/\sqrt{2}) p_2$
3	3	$m_3 = \hat{x} - (1/\sqrt{6}) p_1 - (1/\sqrt{2}) p_2$

The first column gives the region in parity space in which the parity vector is found to lie, as identified by the figure above. The second column shows the instrument whose output would be used. Note that in each case it is the instrument whose measurement axis in parity space is "most normal" to the region where the parity vector lies. The third column shows the estimate expressed in terms of the base vector (the mean of all three measurements) and the (algorithmic) correction which must be made to account for the possibility of instrument failures. On the average, the "best estimate" will be off by the RMS value of this correction. This shows directly how the over-all system performance is degraded by the inclusion of failure tolerance.

#### STATISTICAL FORMULAS

It is interesting to note the simple form taken by statistical formulas in orthogonal parity space when the measurement errors are assumed to be independent, identically-distributed gaussian random variables. Let  $\sigma$  denote the standard deviation of a single measurement error. Then the likelihood function (i.e., the probability density for the observed measurement  $\underline{m}$ , given a particular input  $\underline{x}$ ), is given by

$$P(\underline{m}|\underline{x}) = \frac{1}{(2\pi)^{\ell/2} \sigma^\ell} \exp\left(-\frac{|\underline{p}|^2}{2\sigma^2}\right) \exp\left[-(\underline{x} - \underline{x}_b)^T \mathbf{H}^T \mathbf{H} (\underline{x} - \underline{x}_b)\right]$$

The maximum likelihood estimate of the measured vector  $\underline{x}$  is the base vector  $\underline{x}_b$ , and the likelihood  $P_{\max}$  at this value of  $\underline{x}$  is given by

$$P_{\max} = \frac{1}{(2\pi)^{\ell/2} \sigma^\ell} \exp\left(-\frac{|\underline{p}|^2}{2\sigma^2}\right)$$

The probability density function for the parity vector itself may also be calculated. This probability density is given by the formula

$$P(\underline{p}) = \frac{1}{(2\pi)^{q/2} \sigma^q} \exp\left(-\frac{|\underline{p}|^2}{2\sigma^2}\right)$$

which is the same function except for a multiplicative factor.

This formula implies that the components of the parity vector are independent identically-distributed gaussian random variables with standard deviation  $\sigma$ . Thus, the components of the parity vector have the same probability distributions as the measurement errors.

The square of the length of the parity vector is distributed according to the chi squared probability distribution with  $q$  degrees of freedom.<sup>(14)</sup> In particular, if  $q = 2$ , the probability that the length of the parity vector is less than a fixed value,  $\chi$ , is given by

$$P_r[|\underline{p}| < \chi] = 1 - \exp(-\chi^2/2\sigma^2)$$

so that with ninety percent probability

$$|\underline{p}| < 2.14\sigma$$

#### ESTIMATION ALGORITHMS

By an estimation algorithm we mean simply a computational process to estimate a physical quantity from noisy and possibly faulty measurements. The process may be explicitly or implicitly defined. Thus the "best estimate from a set of good measurements" was defined implicitly above as that  $\underline{x}$  which gives the minimum of a particular function of the measurements, namely the length of the error vector  $|\underline{m} - \mathbf{H}\underline{x}|$ . This then led to the explicit analytical formula,

$$\underline{x} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \underline{m}.$$

In the following we shall formulate an estimation algorithm for the "safest estimate in the presence of possible failures" as the process of finding that  $\underline{x}$  which gives the minimum of some performance index  $L$  which is a measure of the worst error we might make (in estimating  $\underline{x}$ ) when a failure occurs. It will be shown that this algorithm is equivalent to the mid-value selection in the case of three measurements of a scalar quantity.

Unfortunately, it does not appear possible to express the resulting algorithm as an explicit analytical formula in the most general case. However, it will be shown how parity space may be used to simplify the implementation of the computational process.

#### A WORST-CASE PERFORMANCE INDEX

For the moment, in order to simplify notation, let  $\hat{\underline{x}}$  be any estimate that we might make of the "input"  $\underline{x}$ . What we need is some performance index  $L$  which, as a function of the measurement  $\underline{m}$  and the estimate  $\hat{\underline{x}}$ , is a measure of the "worst error" that might have been made by the estimate. What we desire is that this index not depend on any specific (a priori) knowledge about the statistics of the measurement errors.

Now, it would appear that the performance index should be a function of the ratio between some measure of the estimation error and some measure of the input error. It is reasonable to require that they be related, and by using their ratio we can ensure that the estimation error will not be arbitrarily large when all of the measurement errors are small. Of course, some other relationship could be used, but parsimony (and the advantage of hindsight) inclines us toward this choice.

As a measure of the estimation error, it is most natural to use the length  $|\hat{\underline{x}} - \underline{x}|$  of the error itself. However, there are a number of reasonable choices which might be taken as a measure of the error in the measurement, such as the maximum value among the individual errors, and the RSS of the individual errors (that is, the length of the error vector  $|\underline{m} - H\underline{x}|$ ). We shall use the latter for explicitness here, although it is possible to extend the theory by requiring only that the measures chosen have certain abstract properties.

Given these preliminaries, it is reasonable to propose for the performance index which is to be minimized,

$$L(\underline{m}, \hat{\underline{x}}) = \sup_{\underline{x}} \frac{|\hat{\underline{x}} - \underline{x}|}{|\underline{m} - H\underline{x}|}$$

where "sup" means the least upper bound over all  $\underline{x}$ . For the moment we ignore the fact that the denominator may vanish for some measurements  $\underline{m}$  and  $\underline{x}$ . This will happen whenever there is no error in the measurement. We will treat this important case in detail in a separate section below.

Informally, we may interpret the performance index as follows.

Given a measurement  $\underline{m}$  and an estimate  $\hat{\underline{x}}$ , we may suppose that "nature conspires" to have made the actual input  $\underline{x}$  such that we have made the worst possible error in our estimate, relative of course to the actual error in the measurement itself.

If we now require of our estimate  $\hat{\underline{x}}$  that it be chosen for every  $\underline{m}$ , so that  $L_0 = L(\underline{m}, \hat{\underline{x}})$  be a minimum, then we will have

$$|\hat{\underline{x}} - \underline{x}| \leq L_0 |\underline{m} - H\underline{x}|$$

and will have bounded the error in our estimate. Note that here  $\hat{\underline{x}}$  is a value which "realizes" the minimum  $L_0$ . (We will avoid introducing a special symbol to distinguish it from any estimate when there is no danger of confusion.)

Assuming that it is always possible, at least to any required degree of accuracy, to compute a value of  $\hat{\underline{x}}$  which minimizes  $L(\underline{m}, \hat{\underline{x}})$ , then we have implicitly defined the desired estimation algorithm. The set of values  $\hat{\underline{x}}$  which, corresponding to a given  $\underline{m}$ , realize the minimum  $L_0 = L(\underline{m}, \hat{\underline{x}})$  define a mapping  $\hat{\underline{x}} = F(\underline{m})$ . If it could be analytically written out, it would constitute the explicit algorithm discussed above.

## AN OPTIMAL ESTIMATE IN THE ABSENCE OF FAILURES

In order to gain some understanding of the performance index  $L$ , let us first attempt to minimize it on the assumption that there are no failures. We will use orthogonal parity space to simplify the calculations.

Let

$$J(\underline{x}, \hat{\underline{x}}) = \frac{|\hat{\underline{x}} - \underline{x}|}{|\underline{m} - H\underline{x}|}$$

and let us introduce new variables defined by

$$\begin{aligned}\underline{x} &= A\underline{y} + K\underline{m} \\ \hat{\underline{x}} &= A\hat{\underline{y}} + K\underline{m}\end{aligned}$$

where  $K = (H^T H)^{-1} H^T$  and  $A$  is any  $n \times n$  orthogonal matrix, so that  $AA^T = I$ .

Then

$$L(\underline{m}, \hat{\underline{x}}) = \sup_{\underline{x}} J(\underline{x}, \hat{\underline{x}}) = \sup_{\underline{y}} J(\underline{y}, \hat{\underline{y}})$$

To compute the least upper bound on  $J$  we can maximize  $J^2$ .

But

$$J^2 = \frac{(\hat{\underline{x}} - \underline{x})^T (\hat{\underline{x}} - \underline{x})}{(\underline{m} - H\underline{x})^T (\underline{m} - H\underline{x})} = \frac{(\hat{\underline{y}} - \underline{y})^T (\hat{\underline{y}} - \underline{y})}{(\hat{\underline{e}} - H A \underline{y})^T (\hat{\underline{e}} - H A \underline{y})}$$

where  $\hat{\underline{e}} = \underline{m} - HK\underline{m} = V^T V \underline{m}$  is the reduced measurement.

Using the fact that  $H^T \hat{\underline{e}} = H^T V^T V \underline{m} = 0$ , we get easily by direct expansion.

$$(\hat{\underline{e}} - H A \underline{y})^T (\hat{\underline{e}} - H A \underline{y}) = \hat{\underline{e}}^T \hat{\underline{e}} + \underline{y}^T A^T H^T H A \underline{y} \text{ where } \hat{\underline{e}}^2 = |\hat{\underline{e}}|^2.$$

But because  $H^T H$  is an  $n \times n$  real symmetric matrix, it is always possible to choose  $A$  so that  $A^T H^T H A$  is diagonal <sup>(15)</sup>. Thus if  $\lambda_i$  are the eigenvalues of  $H^T H$ , we can write:

$$J^2 = \frac{\sum (\hat{y}_i - y_i)^2}{\hat{\underline{e}}^2 + \sum \lambda_i y_i^2} \quad (\text{sum on } i = 1, \dots, n)$$

Now we observe that the denominator is independent of the signs of the  $y_i$ , but that the numerator will be larger when the signs of the  $y_i$  are taken opposite to those of the corresponding  $\hat{y}_i$ . From this it is clear that whatever value of  $\underline{y}$  is chosen,

$$\sup_{\underline{y}} J(\underline{y}, 0) \leq \sup_{\underline{y}} J(\underline{y}, \hat{\underline{y}})$$

But by the definition of  $\hat{\underline{y}}$ , this is equivalent to saying that whatever the value of  $\hat{\underline{x}}$

$$L(\underline{m}, K\underline{m}) \leq L(\underline{m}, \hat{\underline{x}})$$

We can realize the minimum by taking  $\hat{\underline{x}} = K\underline{m}$ , which is to say, the performance index  $L(\underline{m}, \hat{\underline{x}})$  is minimized for the particular case under consideration by the least squares estimate.

The minimum  $L_0$  is then

$$L_0 = \sup_{\underline{y}} J(\underline{y}, 0) = \sup_{\underline{y}} \left[ \frac{\sum \hat{y}_i^2}{\hat{\underline{e}}^2 + \sum \lambda_i y_i^2} \right]^{1/2} \quad (\text{sum on } i = 1, \dots, n)$$

and it can be shown (we will omit the proof) that this function has a maximum  $L_0 = \frac{1}{\sqrt{\lambda_{\min}}}$  where  $\lambda_{\min} = \min [\lambda_1, \lambda_2, \dots, \lambda_n]$ .



With the above definition, we may now define the fault-tolerant estimation algorithm (that is, fault tolerant to one failure) as the process of selecting that value of  $\hat{x}$  which minimizes  $L(\underline{m}, \hat{x})$ . Unfortunately, the computation of the minimizing estimate  $\hat{x}$  is in general difficult, and an analytic expression for the minimum is not at present known.

Note particularly that the effect of all values of  $i$  on the performance index must be considered in comparing any two values of  $\hat{x}$ : if  $\hat{x}$  is chosen near the value which minimizes  $L_1$  say, then  $L_2$  is likely to be far from its minimum and larger than  $L_1$ , and it is the maximum  $L_i$  which must be taken for  $L$ . For this reason the finally realized minimum value of  $L$  (that is, the value of the estimation error bound) for the fault-tolerant algorithm using  $\ell$  measurements will in general be larger than when the optimal non-fault tolerant algorithm (described previously) is used with  $\ell-1$  measurements.

#### THE ZERO DENOMINATOR PROBLEM

In the definition of the performance indices for both the fault-tolerant and non-fault-tolerant cases, we have ignored the fact that the denominator vanishes for certain values of the variable  $\underline{x}$ . Since it has been assumed that a maximum has been taken with respect to all values of  $\underline{x}$ , it is important that these cases be considered.

So far we have taken as a measure of the measurement errors the length of the error vector  $|\underline{\epsilon}| = |\underline{m} - H\underline{x}|$  or, in the fault-tolerant case, the length of the error vector corresponding to all but one of the measurements,  $\sqrt{|\underline{\epsilon}|^2 - \epsilon_j^2}$ . As mentioned previously, some other measure of measurement error may be desirable in certain cases. Let us generalize the present discussion somewhat by assuming only that the measure is provided by some vector norm  $\|\underline{\epsilon}\|$ , defined as a non-negative number such that, for any vectors  $\underline{x}$ ,  $\underline{y}$  and scalar  $k$ ,

- 1)  $\|\underline{x}\| > 0$  for  $\underline{x} \neq 0$  and  $\|\underline{x}\| = 0$  implies  $\underline{x} = 0$
- 2)  $\|k \underline{x}\| = |k| \|\underline{x}\|$
- 3)  $\|\underline{x} + \underline{y}\| \leq \|\underline{x}\| + \|\underline{y}\|$

We note that these properties hold if  $\|\underline{x}\|$  is defined as the sum of the absolute values of the components of  $\underline{x}$ ; as the length or root-sum-square of the components of  $\underline{x}$ ; or as the maximum absolute value among the components of  $\underline{x}$ . The latter norm will be used in a subsequent example.

The performance indices defined above are specializations of the function

$$L(\underline{m}, \hat{x}) = \sup_{\underline{x}} \frac{|\hat{x} - \underline{x}|}{\|f(\underline{m}, \underline{x})\|}$$

where  $f(\underline{m}, \underline{x})$  is some vector function which is continuous in  $\underline{x}$ . What we really want from a definition of the function  $L(\underline{m}, \hat{x})$  is that given any  $\underline{m}$  and  $\hat{x}$ , the value  $L$  is the least upper bound such that for all  $\underline{x}$ ,

$$|\hat{x} - \underline{x}| \leq L \|f(\underline{m}, \underline{x})\|$$

It is clear that by defining  $L$  as the least upper bound we can ignore the cases when  $f(\underline{m}, \underline{x}) = 0$  since then the right-hand side is independent of the value of  $L$ .

By the use of "sup" in the definitions above we do not mean to imply that the least upper bound is finite. In general this will not be so. If  $\underline{x}_0$  is a solution of the equations  $f(\underline{m}, \underline{x}) = 0$ , then since  $f$  is continuous in  $\underline{x}$  we can make  $\|f(\underline{m}, \underline{x})\|$  arbitrarily close to zero, and the least upper bound is infinite for all  $\hat{x} \neq \underline{x}_0$ ; therefore the least upper bound is realized when  $\hat{x} = \underline{x}_0$ . To be precise we should define

$$L(\underline{m}, \hat{x}) = \sup_{\underline{x} \text{ in } S} \frac{|\hat{x} - \underline{x}|}{\|f(\underline{m}, \underline{x})\|}$$

where  $S$  is (defined to be) the set of all  $\underline{x}$  such that  $f(\underline{m}, \underline{x}) \neq 0$ .

For the non-fault-tolerant case defined above we have  $f(\underline{m}, \underline{x}) = \underline{m} - H\underline{x}$ . This function vanishes only when  $H\underline{x} = \underline{m}$ , which will occur whenever the measurement is without error. Since by definition the rank of  $H$  is equal to the dimension  $n$  of  $\underline{x}$ , there is at most one solution to  $H\underline{x} = \underline{m}$ . Suppose this solution exists and call it  $\underline{x}_0$ . Then  $\underline{y} = \underline{x}_0 - \underline{x}$  gives

$$\begin{aligned} L(\underline{m}, \hat{x}) &= L(\underline{m}, \underline{x}_0) = \sup_{\underline{y} \neq 0} \frac{|\hat{x} - \underline{x}_0 + \underline{y}|}{\|H\underline{y}\|} = \sup_{\underline{y} \neq 0} \frac{|\underline{y}|}{\|H\underline{y}\|} \\ &= \sup_{\substack{\|\underline{u}\|=1 \\ \lambda \neq 0}} \frac{|\lambda \underline{u}|}{\|\lambda H\underline{u}\|} = \sup_{\|\underline{u}\|=1} \frac{1}{\|H\underline{u}\|} \quad (\underline{u} \text{ a unit vector}) \end{aligned}$$

which is clearly finite for any norm.

## AN INTERESTING COMPARISON

The error bound given above,  $L_0 = 1/\sqrt{\lambda \min}$ , was obtained essentially on the basis of a worst-case analysis. Suppose for comparison that the measurement errors are known to be independent, identically-distributed gaussian random variables. Then it is known from statistical estimation theory that the least squares estimate minimizes the RMS estimation error, and that the RMS estimation error is related to the RMS error in the measurements by the formula

$$L_{\text{gaussian}} = \frac{\text{RMS} \left\{ \left| \hat{\underline{x}} - \underline{x} \right| \right\}}{\text{RMS} \left\{ \left| \underline{\epsilon} \right| \right\}} = \sqrt{\frac{1}{\ell}} \sqrt{\text{TRACE} \left\{ (H^T H)^{-1} \right\}}$$

where  $\ell$  is the number of measurements. Some typical values are tabulated below.

Configuration	$H^T H$	$L_0 = 1/\sqrt{\lambda \min}$	$L_{\text{gaussian}}$
Three measurements in one dimension	3	$1/\sqrt{3} = 0.58$	1/3
Orthogonal Triad	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	1	1
Dodecahedron	2 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$1/\sqrt{2} = 0.71$	1/2
Pentad (dodecahedron with one axis removed)	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$	1	$\sqrt{2/5} = 0.63$

## A FAULT-TOLERANT ALGORITHM

The algorithm defined above by the performance index  $L(\underline{m}, \hat{\underline{x}})$  is not fault-tolerant because a failure means one component of  $\underline{\epsilon}$  will be large and thus  $|\underline{\epsilon}|$  will be large. From another viewpoint, the least squares estimate  $\hat{\underline{x}}$  weights all measurements and will be contaminated by the bad measurements.

Now we have defined a failure as a condition where an instrument is contributing more than a tolerable amount of error. If we knew that such a failure had occurred, we would define our performance index only on the measurements from the remaining unfailed instruments. In that case, we would have used for the measure of the measurement error,

$$\min \sqrt{|\underline{\epsilon}|^2 - \epsilon_i^2} \quad (\text{min on } i = 1, \dots, \ell)$$

since this will, in effect, delete the contribution from the instrument with the largest error. Corresponding to the use of this "reduced" configuration, we will have a "partial" performance index:

$$L_i(\underline{m}, \hat{\underline{x}}) = \sup_{\underline{x}} \frac{|\hat{\underline{x}} - \underline{x}|}{\sqrt{|\underline{\epsilon}|^2 - \epsilon_i^2}} \quad (i = 1, \dots, \ell)$$

Then we may say, if we knew that all the instruments were good we would select  $\hat{\underline{x}}$  to minimize  $L(\underline{m}, \hat{\underline{x}})$ . And if we knew that the  $i^{\text{th}}$  instrument had failed, then we would select  $\hat{\underline{x}}$  to minimize  $L_i(\underline{m}, \hat{\underline{x}})$  instead.

However, in general we will not know when an instrument has failed. At best we can attempt to estimate the errors (knowing that we might be using the failure in the estimate), but then we will have to set some thresholds (based on a priori knowledge) in order to decide when a failure had occurred. Clearly, the most pessimistic (and therefore safest) thing to do is simply to select the worst performance index from the set of  $L_i(\underline{m}, \hat{\underline{x}})$ , whether a failure has occurred or not. Thus, we may take for our fault tolerant performance index,

$$\begin{aligned} L(\underline{m}, \hat{\underline{x}}) &= \sup_i L_i(\underline{m}, \hat{\underline{x}}) \\ &= \sup_{\underline{x}, i} \frac{|\hat{\underline{x}} - \underline{x}|}{\sqrt{|\underline{\epsilon}|^2 - \epsilon_i^2}} \quad (i = 1, \dots, \ell) \end{aligned}$$

To summarize, for the non-fault-tolerant performance index, given  $\underline{m}$ :

1) There is exactly one value  $\underline{x}_0$  such that  $H\underline{x}_0 = \underline{m}$  and

$$a) L(\underline{m}, \underline{\hat{x}}) = \infty, \quad \text{if } \underline{\hat{x}} \neq \underline{x}_0$$

$$b) L(\underline{m}, \underline{x}_0) = \sup_{\|\underline{u}\|=1} \frac{1}{\|H\underline{u}\|} \quad (\text{finite})$$

2) There is no value  $\underline{x}_0$  such that  $H\underline{x}_0 = \underline{m}$  and

$$L(\underline{m}, \underline{\hat{x}}) = \sup_{\underline{x}} \frac{\|\underline{\hat{x}} - \underline{x}\|}{\|\underline{m} - H\underline{x}\|}$$

It can be shown that the latter bound is also finite for any norm (we omit the proof). It should be emphasized that in case (1) the algorithm will choose  $\underline{x}_0$  as the "best estimate" since it is the value which minimizes  $L$ ; but this makes sense because  $H\underline{x}_0 = \underline{m}$  says that there is no observable inconsistency in the measurement.

The fault-tolerant case is complicated by the fact that the denominator in the  $L_i$  partial performance indices  $L_i$  can vanish under a number of additional conditions. To simplify discussion, let us introduce  $\ell$  matrices  $P_i$ , each defined as the  $\ell-1$  by  $\ell$  matrix which results from deleting the  $i$ th row of the unit matrix. Then, for any  $\underline{x}$ ,  $P_i \underline{x}$  is the  $\ell-1$  dimensional vector which results from deleting the  $i$ th component of  $\underline{x}$ .

Using the  $P_i$  matrices, the denominators of the  $L_i$  are simply special cases of  $\|f(\underline{m}, \underline{x})\| = \|f(\underline{m}, \underline{x})\|$  with  $f(\underline{m}, \underline{x}) = P_i(\underline{m} - H\underline{x})$ . That is,

$$\|P_i(\underline{m} - H\underline{x})\| = \sqrt{|\epsilon|^2 - \epsilon^2}$$

Following the previous discussion, we are interested in the cases where  $f(\underline{m}, \underline{x}) = P_i(\underline{m} - H\underline{x}) = 0$ . Clearly, when there exists an  $\underline{x}_0$  such that  $H\underline{x}_0 = \underline{m}$ , then the above results apply directly. This will occur not only when the measurement  $\underline{m}$  is without error but also, for one of the  $L_i$  when only the  $i$ th measurement is in error. It is "very nearly" the situation when one of the measurement errors is much larger than the others (the hard failure case). There are also other situations. For example, consider the case of three measurements of a scalar  $x$ , where two of the measurements happen, by an accidental combination of errors, to give the same value. Then

$$\begin{bmatrix} m_1 \\ m_2 \\ m_1 \end{bmatrix} = Hx = \begin{bmatrix} x \\ x \\ x \end{bmatrix} \quad \text{has no solution unless } m_1 = m_2.$$

But

$$P_2 \begin{bmatrix} m_1 \\ m_2 \\ m_1 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_1 \end{bmatrix} = P_2 \begin{bmatrix} x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x \\ x \end{bmatrix} \quad \text{has the solution } x = m_1 \text{ for all } m_2.$$

Suppose that  $\underline{x}_0$  is a value such that  $P_i H\underline{x}_0 = P_i \underline{m}$ . Then  $L_i(\underline{m}, \underline{\hat{x}}) = \infty$  unless  $\underline{\hat{x}} = \underline{x}_0$  and in this case we have, by the same arguments as above

$$L_i(\underline{m}, \underline{x}_0) = \sup_{\|\underline{u}\|=1} \frac{1}{\|P_i H\underline{u}\|}$$

We will assume that  $\ell > n$  (else we would not be able to detect failures at all), that the rank of  $P_i H$  is  $n$ , and  $P_i H\underline{u} \neq 0$  if  $\underline{u} \neq 0$ . Then the denominator is never zero and  $L_i$  is finite.

Note that when  $\underline{m}$  is without error, that an  $\underline{x}_0$  exists (it is the "true value") such that  $P_i(\underline{m} - H\underline{x}_0) = 0$  for each of the partial functions  $L_i$ , and thus each will realize its (finite) least upper bound  $L_i(\underline{m}, \underline{x}_0)$  for the same value  $\underline{\hat{x}} = \underline{x}_0$ . On the other hand, given some other  $\underline{m}$  there may be a solution  $\underline{x}_1$  such that  $P_i(\underline{m} - H\underline{x}_1) = 0$ , and there is no reason to suppose that such a solution would necessarily satisfy  $P_j(\underline{m} - H\underline{x}_1) = 0$  if  $j \neq i$ . But if, for this value of  $\underline{m}$ , there were a value  $\underline{x}_2 \neq \underline{x}_1$  such that  $P_j(\underline{m} - H\underline{x}_2) = 0$ , then since  $L_j(\underline{m}, \underline{\hat{x}})$  would realize its (only) finite value for  $\underline{\hat{x}} = \underline{x}_2$ , we could say for sure that  $L_j(\underline{m}, \underline{x}_1) = \infty$ . And since  $L(\underline{m}, \underline{\hat{x}})$  is defined as the maximum over all the  $L_i$  we would have  $L(\underline{m}, \underline{\hat{x}}) = \infty$  for all values of  $\underline{\hat{x}}$ . For this value of  $\underline{m}$  we would be unable to form a (finite) estimate of the measured value.

Clearly, if we are to have a system which is well defined, then we must require that there be no two values  $\underline{x}_1 \neq \underline{x}_2$  such that for any  $\underline{m}$  and any  $i \neq j$

$$\begin{aligned} P_i H\underline{x}_1 &= P_i \underline{m} \\ \text{and} \quad P_j H\underline{x}_2 &= P_j \underline{m} \end{aligned}$$

Suppose such a case existed. Let  $P_{ij}$  be the  $\ell - 2$  by  $\ell$  matrix which results from deleting the  $i^{\text{th}}$  and  $j^{\text{th}}$  rows of the unit matrix ( $i \neq j$ ). Then  $P_{ij}Hx_1 = P_{ij}\underline{m}$  and  $P_{ij}Hx_2 = P_{ij}\underline{m}$  imply that

$$P_{ij}Hx_1 = P_{ij}\underline{m}, P_{ij}Hx_2 = P_{ij}\underline{m},$$

and therefore  $P_{ij}H(x_1 - x_2) = 0$ . But this implies that  $P_{ij}H$  has rank less than  $n$ , the rank of  $H$ ; and since  $P_{ij}H$  consists of  $\ell - 2$  rows from  $H$ , we can avoid the possibility of this result simply by taking  $\ell - 2 \geq n$ . This extremely important result is not completely unexpected since it takes at least  $n + 2$  measurements to identify a failed measurement, even if the magnitude of the unfailed instrument errors is known.

#### THE ALGORITHM IN PARITY SPACE

The fault-tolerant performance index  $L(\underline{m}, \hat{\underline{x}})$  will first be written in terms of the  $q$ -dimensional parity vector  $\underline{p} = V\underline{m}$  and the  $n$ -dimensional base vector  $\underline{x}_b = K\underline{m}$ . No special assumptions are made about  $K$  except that  $KH = I$  (that is,  $\underline{x}_b$  is not necessarily the least squares estimator). We note that by the considerations of the previous section, we should assume that  $q = \ell - n \geq 2$ . This makes sense physically, because  $q = 1$  would mean that all measurement axes in parity space are parallel and it would therefore be impossible to distinguish between different failures.

For convenience let  $\underline{y} = \underline{x} - \underline{x}_b$ ,  $\hat{\underline{y}} = \hat{\underline{x}} - \underline{x}_b$ .

Then

$$\begin{aligned} L(\underline{m}, \hat{\underline{x}}) &= \sup_{\underline{y}, i} \frac{|\hat{\underline{y}} - \underline{y}|}{\|P_i(\underline{m} - H\underline{x}_b - H\underline{y})\|} \\ &= \sup_{\underline{y}, i} \frac{|\hat{\underline{y}} - \underline{y}|}{\|P_i(\underline{m}\underline{p} - H\underline{y})\|} \end{aligned}$$

In the case where  $|\underline{p}| \neq 0$ , let  $\underline{p} = \mu\underline{u}$  where  $|\underline{u}| = 1$  and  $\mu > 0$ , and define  $\underline{z} = \frac{1}{\mu}\underline{y}$ ,  $\hat{\underline{z}} = \frac{1}{\mu}\hat{\underline{y}}$ .

Then

$$\begin{aligned} L(\underline{m}, \hat{\underline{x}}) &= \sup_{\underline{z}, i} \frac{|\mu(\hat{\underline{z}} - \underline{z})|}{\|\mu P_i(\underline{m}\underline{u} - H\underline{z})\|} \\ &= \sup_{\underline{z}, i} \frac{|\hat{\underline{z}} - \underline{z}|}{\|P_i(\underline{m}\underline{u} - H\underline{z})\|} = L(\underline{m}\underline{u}, \hat{\underline{z}}) \end{aligned}$$

In the case where  $\underline{p} = 0$ , we will have  $\|P_i H\underline{y}\|$  in the denominator, so we will have to take  $\hat{\underline{y}} = 0$ , or  $\hat{\underline{x}} = \underline{x}_b$ .

Then

$$\begin{aligned} L_0 = L(\underline{m}, \underline{x}_b) &= \sup_{\underline{y}, i} \frac{|\underline{y}|}{\|P_i H\underline{y}\|} \\ &= \sup_{\underline{u} = 1} \frac{1}{\|P_i H\underline{u}\|} \end{aligned}$$

By these transformations we see that the performance index can be determined entirely from the  $q$ -dimensional unit vector  $\underline{u}$ , rather than the  $\ell$ -dimensional measurement vector  $\underline{m}$ . This provides, in the fault-tolerant case, a substantial simplification.

For notational convenience let us define

$$\chi(\underline{u}) = \min_{\hat{\underline{z}}} L(\underline{m}\underline{u}, \hat{\underline{z}})$$

The set of  $\hat{\underline{z}}$  which realize the minimum constitute a relation  $\hat{\underline{z}} = \phi(\underline{u})$ . Then we can formulate the fault-tolerant algorithm for the safest estimate  $\underline{x}$  as follows:

- 1) Given  $\underline{m}$  construct  $\underline{p} = V\underline{m}$  and  $\underline{x}_b = K\underline{m}$
- 2) If  $\underline{p} = 0$  then  $\underline{x} = \underline{x}_b$ , and the maximum error in the estimate is  $L_0$ .
- 3) Else construct  $\underline{u} = \frac{1}{|\underline{p}|}\underline{p}$ , set  $\underline{x} = \underline{x}_b + |\underline{p}|\phi(\underline{u})$ , and the maximum error in the estimate is  $\chi(\underline{u})$ .

Of course, in the general case, the computation involved in the above is quite complex. However, in various practical situations it can be greatly simplified. For example, if  $q = 2$ , which is the case with five instruments in 3-dimensional space, the parity vector is 2-dimensional. Thus we can take

$$\underline{u} = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$$

The functions  $\chi(\underline{u})$  and  $\phi(\underline{u})$  can then be expressed in terms of the single variable  $\theta$ , and a practical algorithm can be obtained by expanding the components of  $\phi$  in a Fourier series in  $\theta$ .



## FURTHER EXAMPLES

Consider again the case of three measurements of a scalar quantity  $x$

$$m_1 = x + \epsilon_1$$

$$m_2 = x + \epsilon_2$$

$$m_3 = x + \epsilon_3$$

Let the measure of measurement error be the root-sum-square error

$$\|\underline{\epsilon}\| = |\underline{\epsilon}|$$

then

$$L_1(\hat{x}) = \sup_x \left\{ \frac{|\hat{x} - x|}{\sqrt{(m_1 - x)^2 + (m_2 - x)^2 + (m_3 - x)^2 - (m_1 - x)^2}} \right\}$$

Because of the symmetry of the problem, it is only necessary to evaluate one of the  $L_i$  functions. Thus

$$L_3(\hat{x}) = \sup_x \left\{ \frac{|\hat{x} - x|}{\sqrt{(m_1 - x)^2 + (m_2 - x)^2}} \right\}$$

For convenience, make the change of variables

$$y = x - x_a$$

$$\hat{y} = \hat{x} - x_a$$

where

$$x_a = (m_1 + m_2)/2$$

With these variables, the denominator in  $L_3$  becomes

$$\begin{aligned} (m_1 - x)^2 + (m_2 - x)^2 &= (m_1 - y - x_a)^2 + (m_2 - y - x_a)^2 \\ &= \left( \frac{m_1 - m_2}{2} - y \right)^2 + \left( \frac{m_2 - m_1}{2} - y \right)^2 \\ &= 2(y^2 + d_{12}^2) \end{aligned}$$

where

$$d_{12} = (m_1 - m_2)/2$$

and

$$L_3(\hat{x}) = \sup_y \left\{ \frac{|\hat{y} - y|}{\sqrt{2(y^2 + d_{12}^2)}} \right\}$$

To maximize  $L_3(\hat{x})$ , note that as  $|y| \rightarrow \infty$ , the quantity in braces approaches  $(1/\sqrt{2})$ . Setting the derivative of the quantity in braces to zero yields

$$y = - \frac{d_{12}^2}{\hat{y}}$$

and substituting this value of  $y$  gives for the expression in braces a value of

$$\sqrt{\frac{d_{12}^2 + \hat{y}^2}{2d_{12}^2}}$$

which is  $\geq (1/\sqrt{2})$ .

Therefore

$$L_3(\hat{x}) = \sqrt{\frac{d_{12}^2 + \hat{y}^2}{2d_{12}^2}}$$

or

$$L_3(\hat{x}) = \sqrt{\frac{1}{2} (1 + a_3^2)}$$

where

$$a_3 = \frac{\frac{|\hat{y}|}{|d_{12}|}}{\frac{|\hat{x} - \frac{1}{2}(m_1 + m_2)|}{\left|\frac{m_1 - m_2}{2}\right|}}$$

and by symmetry

$$L_i(\hat{x}) = \sqrt{\frac{1}{2} (1 + a_i^2)}$$

where

$$a_i = \frac{\frac{|\hat{x} - \frac{1}{2}(m_j + m_k)|}{\left|\frac{m_j - m_k}{2}\right|}}{\frac{|\hat{y}|}{|d_{12}|}}$$

and j and k are the other two values of the measurement index.

Since  $L_i(\hat{x})$  is a monotonic function of  $a_i$ , it follows that

$$\begin{aligned} L(\hat{x}) &= \max_{1 \leq i \leq 3} L_i(\hat{x}) \\ &= \sqrt{\frac{1}{2} (1 + \beta^2)} \end{aligned}$$

where

$$\beta(\hat{x}) = \max_{1 \leq i \leq 3} a_i(\hat{x})$$

Finally, since the equations are symmetrical in the measurements, assume that

$$m_1 < m_2 < m_3$$

From the equation for  $a_3(\hat{x})$ , it follows that

$$a_3(m_1) = a_3(m_2) = 1$$

and

$$0 \leq a_3(\hat{x}) \leq 1$$

if

$$m_1 \leq \hat{x} \leq m_2$$

and otherwise

$$a_3(\hat{x}) > 1$$

Similar inequalities hold for  $a_1(\hat{x})$  and  $a_2(\hat{x})$ .

Thus, in the interval  $m_1 \leq \hat{x} \leq m_3$ ,  $a_2(\hat{x})$  satisfies  $a_2(\hat{x}) \leq 1$ , while either  $a_1(\hat{x})$  or  $a_3(\hat{x}) \geq 1$ .

Thus, in this interval,

$$\beta(\hat{x}) = \max \{a_1(\hat{x}), a_3(\hat{x})\}$$

and  $\beta(\hat{x})$  is given by

$$\beta(\hat{x}) = \begin{cases} a_1(\hat{x}) & \text{if } m_1 \leq x \leq m_2 \\ a_3(\hat{x}) & \text{if } m_2 \leq x \leq m_3 \end{cases}$$

or

$$\beta(\hat{x}) = \begin{cases} \frac{m_2 + m_3 - 2\hat{x}}{m_3 - m_2} & \text{if } m_1 \leq x \leq m_2 \\ \frac{2\hat{x} - m_1 - m_2}{m_2 - m_1} & \text{if } m_2 \leq x \leq m_3 \end{cases}$$

or

$$\beta(\hat{x}) = 1 + \begin{cases} \frac{2(m_2 - \hat{x})}{m_3 - m_2} & \text{if } m_1 \leq x \leq m_2 \\ \frac{2(\hat{x} - m_2)}{m_2 - m_1} & \text{if } m_2 \leq x \leq m_3 \end{cases}$$

and over the interval

$$m_1 \leq \hat{x} \leq m_3$$

the minimum value of  $\beta$  is 1, and the minimizing value of  $\hat{x}$  is  $m_2$ .

For  $\hat{x} \geq m_3$ , it follows that

$$a_i(\hat{x}) \geq a_i(m_3) \quad (\text{for } i = 1, 2, 3)$$

and taking the maximum over  $i$ ,

$$\beta(\hat{x}) \geq \beta(m_3) = 1 + \frac{2(m_3 - m_2)}{(m_2 - m_1)} > 1$$

and, for  $\hat{x} < m_1$ ,

$$a_i(\hat{x}) \geq a_i(m_1) \quad (\text{for } i = 1, 2, 3)$$

and

$$\beta(\hat{x}) \geq \beta(m_1) = 1 + \frac{2(m_2 - m_1)}{m_3 - m_2} > 1$$

Therefore, 1 is the minimum value of  $\beta$  and  $m_2$  is the minimizing value of  $\hat{x}$ . Inserting this value of  $\beta$  into the equation  $L(\hat{x}) = \sqrt{\frac{1}{2}(1 + \beta^2)}$ , the minimum value of the performance index is also 1.

Therefore, when the measure of measurement error is the root-sum-square error, the resulting estimate is the mid-value of the measurements, and the estimation error is always less than or equal to the root-sum-square of the two smallest measurement errors.

Note that when

$$\hat{x} = m_2$$

the components of the performance index have the values

$$L_1(\hat{x}) = 1$$

$$L_2(\hat{x}) < 1$$

$$L_3(\hat{x}) = 1$$

and the estimator is trading-off the penalty due to a failure in  $m_1$  against the penalty due to a failure in  $m_3$ .

As a second example, again consider the case of three measurements of a scalar quantity, but let the measure of measurement error be the maximum measurement error

$$\|\epsilon\| = \max_{1 \leq i \leq 3} |\epsilon_i|$$

Again, by symmetry, it is only necessary to evaluate one of the components of the performance index. Employing the notation of the preceding example,

$$\begin{aligned} L_3(\hat{x}) &= \sup_x \left\{ \frac{|\hat{x} - x|}{\max(|m_1 - x|, |m_2 - x|)} \right\} \\ &= \sup_y \left\{ \frac{|\hat{y} - y|}{\max(|y - d_{12}|, |y + d_{12}|)} \right\} \\ &= \sup_y \left\{ \frac{|\hat{y} - y|}{|y| + |d_{12}|} \right\} \\ &= \sup_y \left\{ \frac{|\hat{y}| + |y|}{|y| + |d_{12}|} \right\} \end{aligned}$$

Now

$$\frac{|\hat{y}| + |y|}{|y| + |d_{12}|} = 1 + \frac{|\hat{y}| - |d_{12}|}{|d_{12}| + |y|}$$

and if

$$|\hat{y}| \leq |d_{12}|$$

$L_3(\hat{x})$  is maximized by letting  $|y| \rightarrow \infty$ , yielding

$$L_3(\hat{x}) = 1$$

On the other hand, if

$$|\hat{y}| > |d_{12}|$$

$L_3(\hat{x})$  is maximized by taking  $|y| = 0$ , and

$$L_3(\hat{x}) = \frac{|\hat{y}|}{|d_{12}|}$$

or, in either case

$$\begin{aligned} L_3(\hat{x}) &= \max \left( 1, \frac{|\hat{y}|}{|d_{12}|} \right) \\ &= \max \left( 1, a_3(\hat{x}) \right) \end{aligned}$$



Thus, by symmetry

$$L_i(\hat{x}) = \max [1, a_i(\hat{x})]$$

and maximizing over  $i$ ,

$$L(\hat{x}) = \max [1, \beta(\hat{x})]$$

It was shown in the preceding example that

$$\beta(\hat{x}) > 1$$

unless

$$\hat{x} = m_2$$

and that

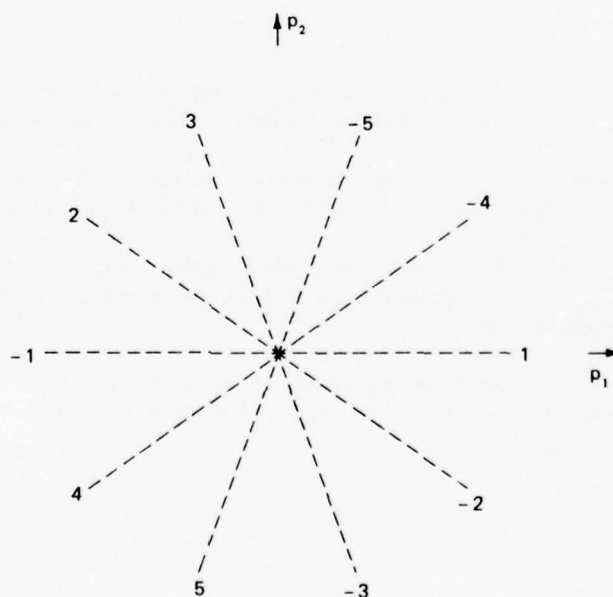
$$\beta(m_2) = 1$$

Again the resulting estimate is the mid-value of the measurements and the minimum value of the performance index is 1. In this case, the unit minimum value of the performance index implies that the estimation error is always less than or equal to the second largest measurement error. At the minimizing value of  $\hat{x}$ , the components of the performance index have the values

$$L_1(\hat{x}) = L_2(\hat{x}) = L_3(\hat{x}) = 1$$

#### SOME NUMERICAL RESULTS ON THE PENTAD

The pentad defined in the examples above is particularly interesting because it is a practical configuration of the minimum number of instruments needed to implement the algorithmic covering technique (that is, total failure detection by output comparison) in 3-dimensional space. In this case the parity space is 2-dimensional, and a number of symmetries exist which can be used to further simplify the computational problem. When the parity vectors corresponding to the individual measurement axes are plotted, we find that there is an axis every 36 degrees.



Consider the special case where the measure of measurement error is taken as the maximum absolute value among the component measurement errors:

Because of the symmetries in this problem, it is possible to show (we omit the proof) that if  $\underline{p}$  is expressed as

$$\underline{p} = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} \quad (0 \leq \theta \leq 360^\circ)$$

then it is only necessary to examine the values of  $\theta$  such that  $0 \leq \theta \leq 18^\circ$ . It is fairly straightforward to organize the computation of  $L(\underline{m}, \hat{\underline{x}})$  into choosing alternatives from a finite set of computable simple functions, and then to find an approximation to  $L_{\min}$  by a minimum seeking computational process. Some approximate typical results are

$\theta$	$L_{\min}$	
$0^\circ$	5.33	(zero denominator case)
$6^\circ$	5.33	
$12^\circ$	5.33	
$18^\circ$	5.33	

It appears that  $L_{\min}$  is nearly constant (the above results were computed to a number of additional decimal places). A speculation is that it may be independent of  $\theta$  for this case.\* Thus, the maximum RSS estimation error is always less than about 5.33 times the second largest measurement error for this instrument array.

This number may be compared with the value 4.80 which is the ratio of the maximum RSS estimation error for the bounding sphere estimate<sup>(11)</sup> to the maximum error assumed for an unfailed instrument in the presence of a single failure. The bounding sphere estimate is optimal in the sense that it minimizes the ratio of the maximum RSS estimation error to the maximum error for unfailed instruments when a hard upper bound on errors of unfailed instruments is known. If there is a hard upper bound on unfailed instrument errors, the present estimate will have an estimation error bounded by 5.33 times the unfailed instrument error bound in the case of a single failure, since the second largest instrument error will be less than the unfailed instrument error bound. Thus, we are paying an estimated error penalty of  $5.33 - 4.80 = 0.53$  times the maximum unfailed instrument error because we are assuming that this error bound is unknown. Note that the 4.80 bound for the bounding sphere estimate was obtained by a Monte-Carlo simulation of measurement errors. Since these may not have exactly sampled the worst-case set of measurement errors, the true bound for the bounding spheres estimate may be slightly higher than the 4.80 figure.

The components of the vector  $\underline{\phi}(\theta)$  defined in the previous section, are corrections to the estimate to account for failure tolerance. For the angles  $0^\circ$ ,  $6^\circ$ ,  $12^\circ$  and  $18^\circ$ , they were found to be linear functions of  $\theta$  to an accuracy of a few percent. Thus it appears possible to implement the fault-tolerant algorithm for the symmetrical pentad with very little computational complexity.

#### SUMMARY

The above discussion covers many points which are important in the design and implementation of redundant instrument systems, and provides a basis for further work on practical thresholdless fault-tolerance. Two important results are:

- (1) A worst-case performance index is developed which requires no knowledge about the statistics of the instrument errors. It is shown that the least squares estimator minimizes this index, but that the resulting algorithm is not fault-tolerant.
- (2) A fault-tolerant algorithm is then defined which uses the worst-case performance index to bound the estimation error by the product of the index and the RSS of the individual instrument errors with the largest error ignored: the algorithm chooses as the estimate the value which minimizes the worst-case index.

The important point to emphasize is that fault-tolerance is achieved without the need to determine any thresholds, and without having to identify when or in which instrument the failure occurs. Numerical results indicate that useful error bounds are realized.

\* Since preparation of this paper, an exact formula for this bound has been found.

## LIST OF REFERENCES

- 1 Ephgrave, J.T., "Optimum Redundant Configurations of Inertial Sensors", Report No. TOR-1001 (9990)-5, Sept. 1966, AEROSPACE CORPORATION, El Segundo, CA.
- 2 Bejczy, A.K., "Non-Orthogonal Redundant Configurations of Single-Axis Strapped-Down Gyros", JPL Quarterly Technical Review, Vol. 1, No. 2, July 1971, pp. 107-118.
- 3 Pejsa, Arthur, J., "Optimum Orientation and Accuracy of Redundant Sensor Arrays", AIAA Paper No. 71-59, Jan. 1971, New York, N.Y.
- 4 Gilmore, Jerold P., and Richard A. McKern, "A Strapdown Inertial Reference Unit (SIRU)", Journal of Spacecraft and Rockets, Vol. 9, No. 1, Jan. 1972, pp. 39-47.
- 5 Giardina, Charles R., "Optimal Gyro Mounting Configurations for a Strapdown System", NAECON '74 RECORD, pp. 584-591.
- 6 Fraser, D.C. et al., "First Quarterly Progress Report", Contract NAS 9-4065, Task Order 41, Aug. 30, 1971, Charles Stark Draper Lab, MIT, Cambridge, Mass., Chap. III.
- 7 Ephgrave, J.T., "Redundant Adaptive Strapdown Inertial Navigation System", Report TOR-0066 (5306)-10, Oct. 1969, AEROSPACE CORPORATION, El Segundo, CA.
- 8 Evans, F.A. and Wilcox, J.C., "Experimental Strapdown Redundant Sensor Inertial Navigation System", Journal of Spacecraft and Rockets, Vol. 7, No. 9, Sept. 1970, pp. 1070-1074.
- 9 Chien, T.-T., "An Adaptive Technique for a Redundant-Sensor Navigation System", AIAA Paper 72-863, Stanford, CA, 1972.
- 10 Wilcox, J.C., "Maximum Likelihood Failure Detection for Redundant Inertial Instruments", AIAA Paper 72-864, Stanford, CA., 1972.
- 11 Potter, James E., and James C. Deckert, "Minimax Failure Detection and Identification in Redundant Gyro and Accelerometer Systems", Journal of Spacecraft and Rockets, Vol. 10, No. 4, April 1973, pp. 236-243.
- 12 Wilcox, James C., "Retroactive Failure Correction for Strapdown Redundant Inertial Instruments", Journal of Spacecraft and Rockets, Vol. 12, No. 6, June 1975.
- 13 Wilcox, J.C., "Competitive Evaluation of Failure Detection Algorithms for Strapdown Redundant Inertial Instruments", Journal of Spacecraft and Rockets, Vol. 11, July 1974, pp. 525-530.
- 14 Box, J.E.P. and G.M. Jenkins, "Time Series Analysis," Holden-Day, c. 1970, p. 259.
- 15 Noble, Ben, "Applied Linear Algebra", Prentice-Hall, Inc., c. 1969, p. 388.

## TIME - DIVISION MULTIPLEXED DATA BUS INTEGRATION TECHNIQUES

Erwin C. Gangl  
 Technical Advisor  
 Digital Avionics/Standardization  
 ASD/ENAI  
 Wright-Patterson AFB, OH 45433

## SUMMARY

Today, avionics are demanding an increasing proportion of the resources available for aircraft weapons systems. These avionics are providing increased capability and accuracy to the aircraft weapon system; but, also are a prime contributor to increased complexity and decreased reliability of the system. Digital avionics appears to offer the desired increase in capabilities and performance without the normal companions of low reliability, complexity, and high cost because it is amenable to mechanization via solid state devices; it is more orderly and systematic, and provides growth and change without major hardware modification. Digital avionic integration, in order to reap these benefits, requires standard equipment interfaces and a standard approach to data intercommunication. The time division data bus is the technique that permits this new concept of system integration to emerge. This paper will present the data bus evolutions, its standardization and application. The acquisition management and logistic benefits will be discussed.

## DEFINITIONS

1. Time Division Multiplexed Data Bus: Throughout this paper, there are many shortened versions of it: multiplex data bus, data bus, bus, multiplexing, MUX and MIL-STD-1553.
2. Time Division Multiplexing (TDM): The transmission of information from several signal channels through one communication system with different channel samples staggered in time to form a composite pulse train.
3. Remote Terminal: This is the electronics necessary to interface the bus with the sub-system and the sub-system with the bus.
4. Bus Controller: The controller shall be a unit that is either programmable, or controlled by a processor, and that serves the function of commanding, scanning and monitoring bus traffic.
5. Message: A message is a transmission of words on the data bus cable. A message transfer is complete when the command word, data word(s) and the status word have been transmitted. There are three types of messages: controller to terminal, terminal to controller and terminal to terminal.
6. Words in a Message: In this document a word is a sequence of 16 bits plus sync and parity. There are three types of words: Command, Status and Data.

## INTRODUCTION

Current US Air Force avionics acquisition practices breed "black box" proliferation resulting in high cost, low reliability, and a heavy operating and maintenance burden.

A study<sup>1,2</sup> was conducted to investigate the applicability of digital techniques to solve today's high cost and proliferation of aircraft avionics. It was found that in current aircraft, the avionics cost is approximately one-third of the total system acquisition cost. Because of the lack of a "standard" integration approach, equipment introduced into inventory is generally in low quantity and of low reliability and, therefore, results in a logistic and maintenance nightmare. Even equipment intentionally starting out identical, ends up unique because of the different interface requirements in different systems, causing them to become incompatible.

One solution proposed by the study group was to view the aircraft avionics as an "integrated system" rather than a conglomerate of functional sensors. The data bus concept forces one to perform systems level analysis because information flow definition is used as the key element in the orderly, standard integration process of avionic equipment (black boxes).

Digital avionics will be used more and more in current and future aircraft systems and the data bus will be the key to successful integration. Other requirements are sensors built to the bus' standard digital interface, the liberal use of common digital computers, modular/multipurpose controls and displays and a standard higher order software language. When utilizing the data bus concept, it is important to realize that a large amount of the integration is accomplished through software. Therefore, the final and most critical integration step is performed through the effective application of flight software; i.e., the software controls the real-time information flow.



## THE DATA BUS

The digital time-division multiplex data bus is a tool to aid in system integration. Originally introduced to save avionic hardware interconnect wiring weight and ease computer interface requirements, it is now considered the cornerstone of digital avionics. Through the use of the multiplexing technique, many other benefits can be reaped. The resultant standard electronic interface permits a building block approach to systems architecting and integration. Retrofits become easier, standard interchangeable sensors become a reality, and maintenance is greatly improved. Flexibility is probably the data bus' greatest attribute, reliability, built-in-test, redundancy and graceful degradation are additional benefits. These gains are realizable only if some form of standardization is adhered to; therefore, a military standard was developed to provide some consistency in data bus design, MIL-STD-1553A<sup>3,4</sup>. This standard has been used in a number of Air Force and Navy programs. Many new applications are developing. DAIS<sup>5</sup> is a laboratory program to help define and explore applications of digital avionics. The Air Force is committed to the application of digital avionic techniques and the multiplexed data bus concept is an important part of it.

## BACKGROUND

Typical signal integration is difficult because sensor/equipment interfaces are unique (non-standard signal characteristics) resulting in a significant weight and volume penalty in large, avionic dispersed aircraft systems. Each sub-system has its own unique input/output signalling formats, from non-standard analog, synchro, discrete, digital serial and parallel, to infinite combination of these.

The majority of avionic sensors are designed to provide data in an analog form most convenient to the sensor manufacturer or peculiar to a one-time application. Therefore, the computer contractor, to properly communicate with the sub-systems, must build a special purpose converter unit to interface the computer with these sub-systems. Each analog or discrete signal is routed separately from each sub-system to this central converter. The converter unit provides such things as signal terminations, conditioning, sampling, scaling and conversion to the proper digital format. The analog to digital and digital to analog converters must be extremely high speed because they are time shared among all the input/output signals. Often this converter unit is twice as complex as the computer and highly special purpose. That is, if any signal format is changed or a new one added, there is a definite impact on the converter hardware. Changes are often costly and major avionic modifications impossible without starting from scratch.

A digital avionic multiplex data bus has a number of definite advantages. Substantial wire savings can be realized by using the same data bus in a time sharing mode. Wire and connector savings show up as weight savings. Ease of changing sensors due to standard interfaces is another extremely desirable feature. The digital transmission technique used is less susceptible to EMI and it is easier to detect and correct errors. System configuration modifications can be performed by properly changing the computer software. Computers will no longer become obsolete because they will be truly general purpose (i.e., special purpose converter hardware will no longer be an integral part of it). In this digital bus concept, sometimes called MUX for short, the sensors that provide real time data to the computer or receive data from it are all tied in parallel to a common multiplexed data bus. Transmission is digital under central computer control. This requires federated conversion and federated data storage in a sensor scratch pad memory. The sensor generates the data, converts it and stores it in its own scratch pad (remote data memory) at its own iteration rate. The central computer, under software control, samples these data memories as required by the operational program and treats them as if they were an extension of the central computer's main memory. Data gathering is accomplished through the use of the computer programmed input/output controller.

An interesting advantage to this technique is that the data transfer rates on the bus are much lower, requiring a much lower bandwidth transmission line. A one megahertz data transmission rate is more than sufficient. A data word is transferred only when needed rather than continuously as in the analog. (Continuous transmission is required in the analog transmission technique because the sensor and converter hardware never know when and even if the software requires the newly generated information.) Consequently, the computer memory must always be updated with the most current data. In the multiplex data bus technique, the sensor scratch pad memories are treated as an extension of the computer's main memory and their use is under program control. Therefore, additions and deletions of sensors are easily accomplished. Sensor modifications that change the sensor's analog input/output format will require the sensor contractor to modify his converter and it will be totally his responsibility to comply. These modifications will then be performed by the contractor in the sensors own lower speed converter unit requiring only software modifications within the central computer.

## SYSTEM APPLICATIONS

The first systems application of this type of a multiplexed data bus was in the integration of the USAF's F-15 fighter aircraft. (Figure 1). Some of the reasons why the data bus was originally used were:

1. Weight savings (wire, connectors)

2. Simpler wire routing in aircraft (fewer bulkhead holes, clamps, connectors)
3. Fewer data transfers (sensor information transmitted on demand only)
4. Ease of changing sensors (made possible because of the resulting standard interface)
5. Less tendency to obsolescence (sensors, computers)
6. Less electromagnetic interference (because of digital transmission)
7. Higher reliability (better error detection, fewer hardwire interconnections)
8. Permits retrofits (eases modification and growth problems)
9. Lends itself to simplified built-in-test (BIT) techniques

Once proven feasible, another major USAF system, the B-1 strategic bomber, designed its avionic system around the data bus. (Figure 2). Three separate bus systems are utilized in the B-1 avionic systems.

The first one, quad redundant, is used for electrical power switching control and management (EMUX); the second, dual redundant, for mission avionic sensor integration, both offensive and defensive (AMUX), and the third, no redundancy, for built-in-test purposes called "Centrally Integrated Test System" (CITS).

At this point in time, an undesirable trend was perceived in the development of multiplex data bus (MUX) techniques, namely that MUX systems were beginning to proliferate to an extent which was threatening to obviate the gains which they had provided as noted above. A number of different MUX buses had been built, all of which possessed similar architectures, data rates, encoding techniques, and technology. However, there was sufficient difference in their signal formats and protocol, interfaces, and functional operation to preclude any common hardware or interfaces. Further, it was apparent that the use of MUX techniques was spreading into other non-traditional avionic areas such as engine management, stores management, flight instrumentation and flight controls. In summary, the time was ripe for standardization of MUX systems.

Drawing on the F-15 and B-1 experience, and reviewing anticipated future avionics needs, a draft standard was prepared that resulted in MIL-STD-1553 (USAF), 30 Aug 1973. Resultant tri-service negotiations, based on US Navy and Army application requirements, caused the standard to be upgraded and it was published as a tri-service document, MIL-STD-1553A, on 30 April 1975.

#### THE MILITARY BUS STANDARD

MIL-STD-1553A, entitled "Aircraft Internal Command/Response Time Division Multiplex Data Bus", covers the overall systems requirements, architecture, operational protocol, and interfaces within the multiplexed data bus system. It is intended to establish uniform requirements for all multiplex applications, not just avionics, and provides for commonality of electronic functions and interfaces within the total air vehicle. The standard was written so as to permit the system designer a maximum amount of design flexibility while retaining a common inter-weapon system interface. As a result, MIL-STD-1553A must be employed in conjunction with detailed air vehicle or subsystem specifications.

Figure 3 illustrates an elemental bus configuration using MIL-STD-1553A. The bus system has three major components, the bus controller, the data bus itself, and remote terminals or subsystems. The bus controller acts as the focal point for the data bus and the integration of the subsystems connected to the bus. The bus controller is a software programmable device, a computer or dedicated microprocessor, which issues the commands to initiate data transfers over the data bus. No remote terminal transmits a signal without the transfer being initiated by the controller; its commands, and the remote terminals respond, hence the description "command/response" architecture. When growth is required in the system, the new subsystem is added and given a unique address. The only change required to the existing equipment is a modification to bus controller software, thus providing a high level of flexibility without major equipment modifications.

The data bus itself is a shielded twisted wire pair, operating in a classical balanced transmission line mode. Redundancy (i.e., multiple data buses) is the system designers option. (Figure 4). Transformer coupling is used between the bus and the terminals; the signal on the bus when transmitting data is a serial bit stream, Manchester Bi-Phase Level encoded, and transmitted at a 1.0 MHz rate.

The remote terminals provide that hardware necessary to interface the subsystem to the bus; this includes transmitter/receiver, encoder/decoder, error checking, and holding registers. The remote terminal may exist as a separate line replaceable unit (LRU), or be built into a subsystem which interfaces to the data bus (see Figure 3). In general, any moderately complex subsystem (e.g., an inertial subsystem) will interface directly to the bus, whereas the relatively simple or low data rate subsystems (e.g., a TACAN or doppler altimeter) would interface to the bus through a remote terminal which can handle more than one of these.

Three types of words are used to make up the messages which are transmitted between subsystems on the data bus; Command, Data and Status words as shown in Figure 5. Command words are only transmitted by the bus controller, Status words only by remote terminals, and Data words by each. All words are preceded by a unique synchronization waveform which is distinguishable from the remainder of the Manchester encoded bits. The Command word is divided into four fields; the first being the address of the terminal to which the Command word is directed; second, the transmit/receive bit which indicates the remote terminals action; third, a subaddress/mode field to be defined by the remote terminal designer; and fourth, a count of the number of words to be transferred. The last bit in the word is used to provide odd parity over the preceding sixteen bits.

The Data word, which contains the information to be transferred, has one sixteen bit field, followed by parity.

The Status word, which is always transmitted by the terminal in response to a command is also partitioned into four fields. First is the terminal address; second, a message error bit to indicate the failure of the preceding message (e.g., a parity error in one of the preceding words in the message); third, nine bits which can be used for defining status codes; and fourth the terminal flag bit which indicates the need for the bus controller to request status and self-test information from the terminal. Parity is also provided for the Status word.

The Command, Data, and Status words are combined to form three message formats as shown in Figure 6. A bus controller to remote terminal transfer is accomplished by the controller sending a receive Command word followed contiguously by the Data words. After a specified gap time, the addressed terminal responds with its Status word, thus indicating proper receipt of the message.

A terminal to controller transfer is initiated by the controller sending a transmit Command word to the remote terminal. After the gap time, the addressed terminal responds with a Status word contiguously followed by the specified number of Data words.

The direct terminal-to-terminal transfer may be effected by a combination of the two previous commands. It is important to note that continuous "handshaking" between remote terminals and the bus controller provides a constant system health monitoring function, thus providing an inherent built-in-test (BIT) capability of the MUX bus.

The system designer using MIL-STD-1553A still retains a high degree of flexibility in his design approach due to the nature of the standard. His options include selecting the redundancy scheme (dual, triple, or quad? - Figure 4) determining the bus controller implementation, identifying criteria for selection of a signal for transmission on the bus, defining the physical configuration of the bus system, and providing functional operation algorithms. Due to its flexibility, MIL-STD-1553A has received wide acceptance by industry and DoD, and has already been applied to a large number of military weapon systems.

#### APPLICATION BENEFITS OF THE DATA BUS

MIL-STD-1553A is being applied to a variety of current weapon systems, both as an integration tool to reduce aircraft wiring and to increase reliability and flexibility, and also, as a standard digital interface to subsystems. The following systems are using, or contemplating using MIL-STD-1553A; F-16 (Figure 7), F-18, DAIS, LAMPS, RPV, AMST and Army helicopters, ATF. NATO AGARD<sup>6</sup> is investigating commonality between international multiplex data buses.

If multiplexing is used as an integration technique in weapon systems, great standardization benefits can be realized. Multiplexing defines the way information flows on the data bus and also the protocol needed to accomplish orderly data transfers. This enables one to define two standard interfaces; one to the avionic subsystems that have digital/discrete inputs/outputs and the other directly to the data bus (twisted pair) for subsystems that have the multiplexed remote terminals built in (Figure 3). All this is defined in MIL-STD-1553A.

Other standardization tasks are now underway in the area of airborne computers and its application and support software, multipurpose controls and displays and sensors with standard interfaces. Because digital techniques are expanding into areas that were formerly purely electrical and/or mechanical in nature (i.e., electrical power control and flight control, etc.) the AF Digital Avionics Study<sup>1</sup> has proposed to change the definition of "avionics" to "airborne electronics". This is why processors, software and multiplexing are appearing in so many new disciplines. Because not all the electronics for these disciplines are always procured simultaneously, much proliferation can occur in the air vehicle hardware. That is why multiplexing is so readily accepted. It gives the systems engineer a building block approach to system design and permits standardization throughout the total weapons system. Subsystems not yet available can easily be simulated and then substituted later when delivered, i.e., systems integration is no longer delayed until all parts are delivered.

Another benefit derived from the MUX standard is that it permits one to establish an in-house systems engineering capability. In order to get a better handle on avionics, one can investigate the feasibility of performing systems analysis in-house, thus accomplishing your own top level/functional systems definition and architecting. This type of system simulation allows one to investigate the data flow correctness and systems



performance based on data accuracy, timing and availability. Redundancy and graceful degradation can be realistically simulated to find out critical parameters and system sensitivity to various signals and their accuracy. This approach to systems integration/architecting will reduce cost by avoiding over-specification of equipment, it reduces risk through early systems simulation prior to specification writing, and it increases reliability through performing failure analysis and applying well defined built-in-test (BIT) and graceful degradation/redundancy schemes.

Other benefits that multiplexing provides can be realized in system retrofits and in the maintenance and logistics area. Systems retrofits are more easily accomplished because of the standard interfaces of subsystems and time sharing of aircraft wiring (i.e., MUX Bus). In the maintenance area, the data bus enables dynamic testing of on-board equipment and standard ground support equipment. In the logistics area the prime benefit is fewer inventory parts (at all levels) and increased reliability of the hardware.

In summary, to reiterate, by first standardizing the multiplexed data bus (MIL-STD-1553A), and following it up with similar standards that are developed for reducing proliferation in computers and software, the following overall benefits can be gained:

1. Systems integration/architecting will be simplified.
2. Standardization of interfaces, hardware and software will reduce proliferation.
3. Retrofit and maintenance of digital avionic systems will become easier.
4. Reliability of hardware will be increased and the resultant reduction of inventory equipment, parts and documentation will reduce life cycle costs.
5. There will be more international competition in avionics hardware/software because of the standard/compatible/interchangeable sensors that can be developed off-line.

Therefore, it is felt that even though the multiplex data bus in only a portion of the overall digital avionics concept, it has laid the cornerstone for the concept by providing a standard integration concept with standard equipment interfaces that is reconfigurable and technology independent and as a result the many benefits listed in this paper can be realized.

#### REFERENCES

1. Bright, B., et al                      USAF, Aircraft Avionics (Digital Avionics Study Report), April 1973, ASD-TR-73-18, Volume 1
2. Gangl, Erwin C.                      "Air Force Findings and Recommendations on Digital Aircraft Avionics", NAECON 1974, pp 72-77
3. \_\_\_\_\_                      "A Tri-Service Military Standard for Aircraft Internal Time Division Command/Response Multiplex Data Bus", MIL-STD-1553A, 30 April 1975  
(Supersedes MIL-STD-1553 (USAF), 30 Aug 1973)
4. Gifford, Charles A.                      "A Military Standard for Multiplex Data Bus", NAECON 1974, pp 85-88
5. List, Bernie                      "DAIS: A Major Crossroad in the Development of Avionics Systems", Astronautics, Jan 1973, pp 55-61
6. Schweizer, G., et al                      "A Study of Standardization Methods for Digital Guidance and Control Systems", AGARD Advisory Report No. 90, 1976



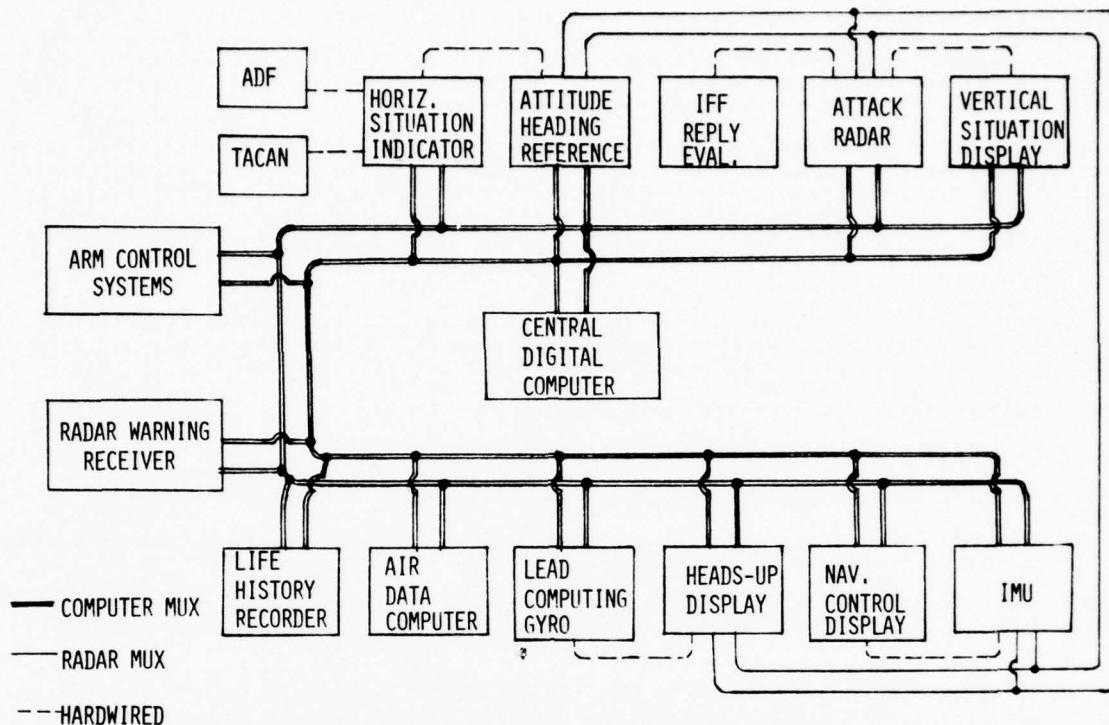


Fig. 1 F-15 Digital System Architecture

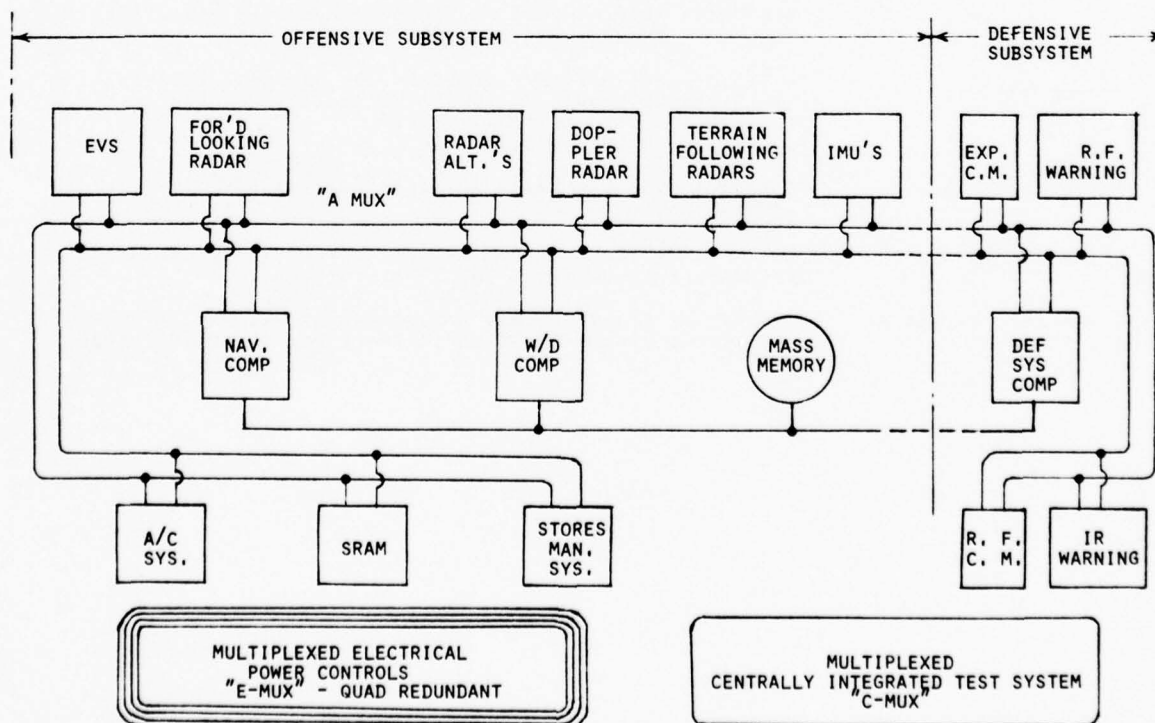


Fig. 2 B-1 Digital System Architecture

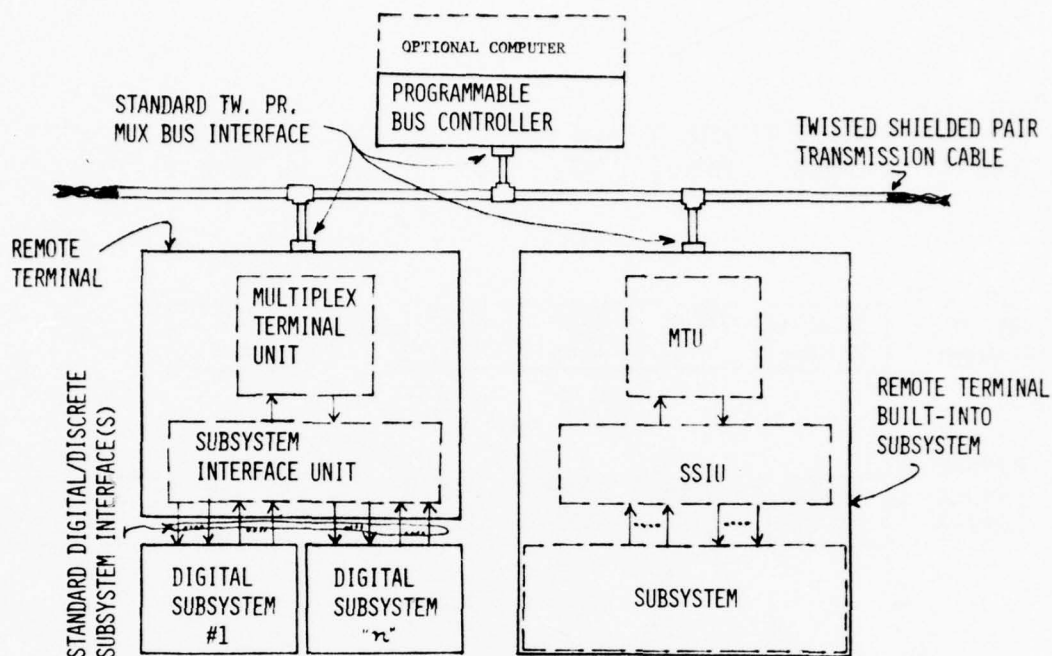


Fig. 3 Elemental Bus Architecture

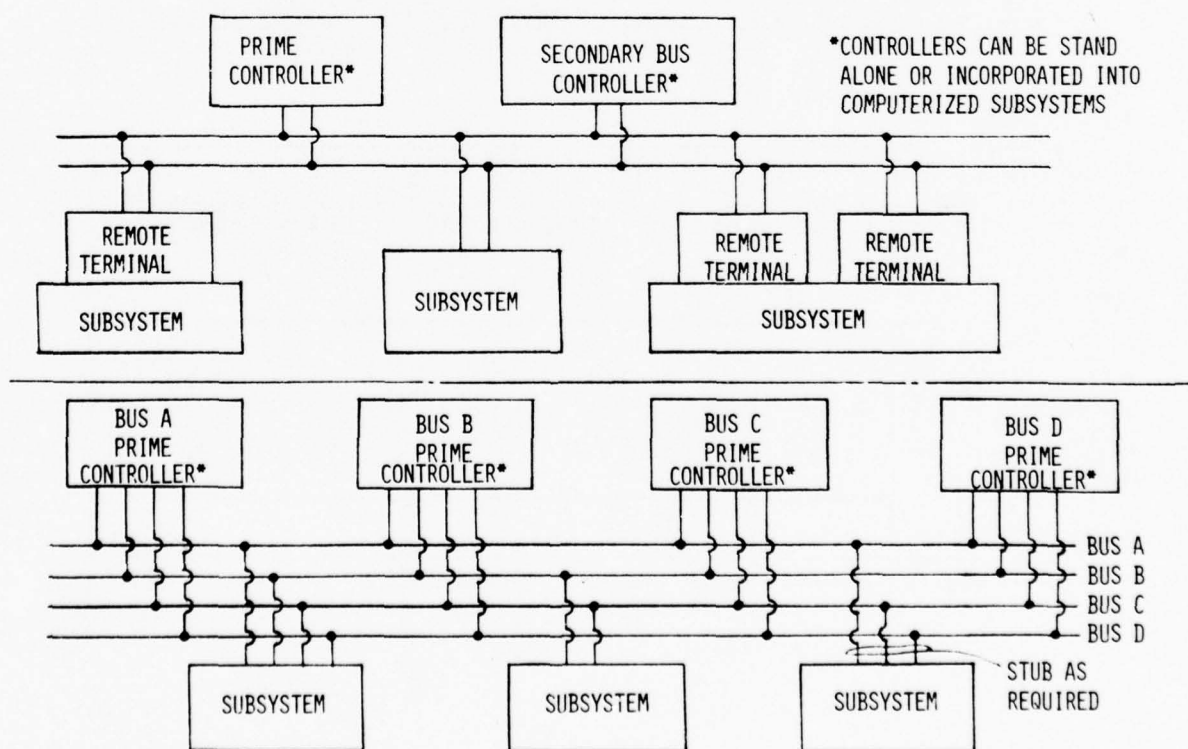


Fig. 4 Complex Bus Architectures

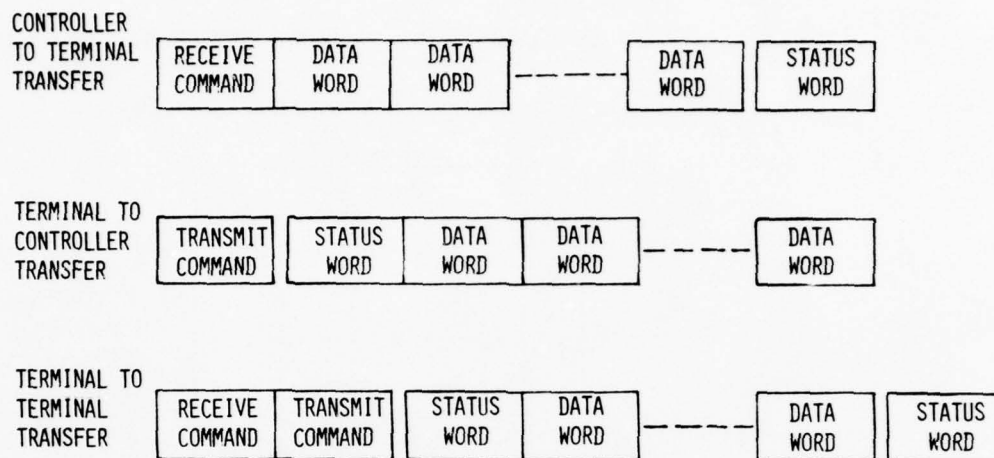
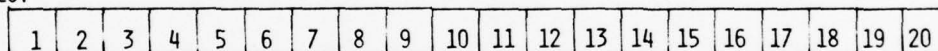
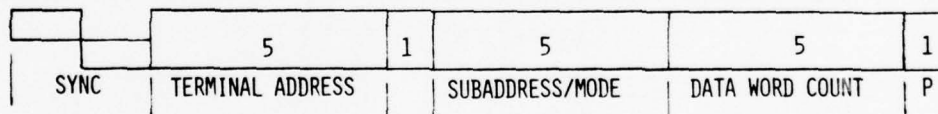


Fig. 5 Message Formats

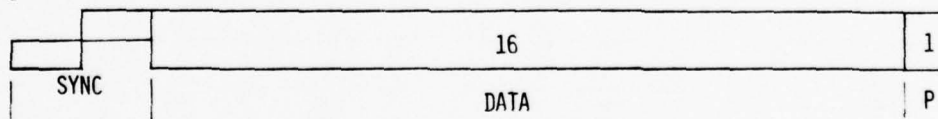
BIT TIMES:



COMMAND WORD:



DATA WORD



STATUS WORD:

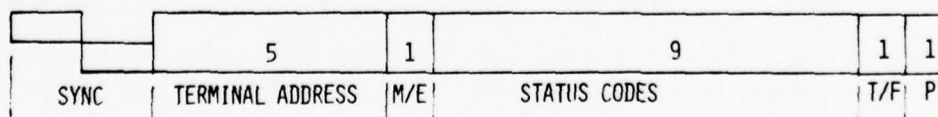


Fig. 6 Word Formats

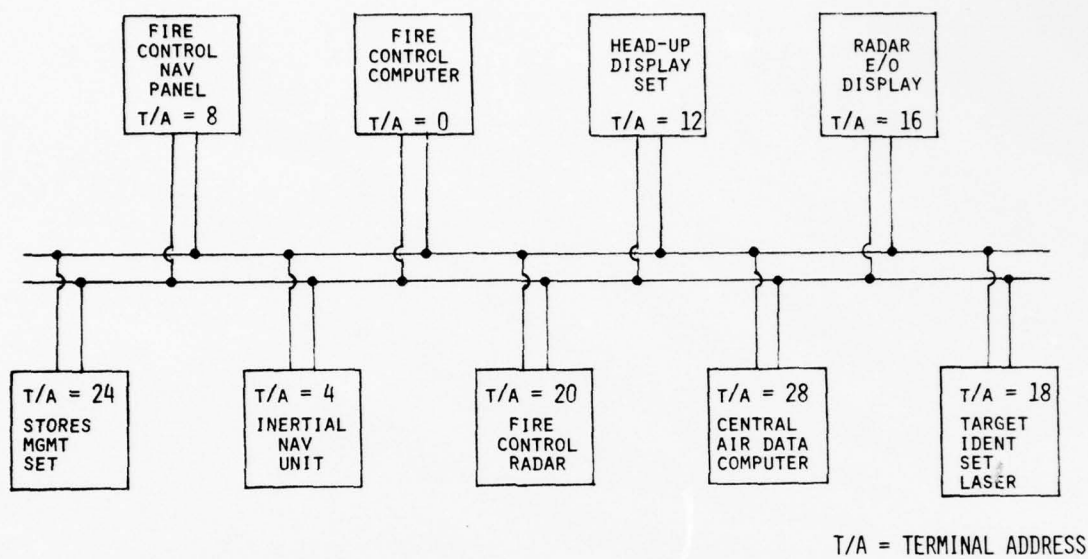


Fig. 7 F-16 Multiplex Systems Architecture



## HIGHLY RELIABLE MULTIPROCESSORS

By

Nicholas D. Murray  
NASA/Langley Research Center  
Hampton, Virginia 23665

Albert L. Hopkins  
The Charles Stark Draper Laboratory  
Cambridge, Massachusetts 02139

John H. Wensley  
Stanford Research Institute  
Menlo Park, California 94025

SUMMARY

This paper discusses highly reliable fault-tolerant computer systems for use in flight-critical avionic and control systems of future commercial transport aircraft. Such aircraft are envisioned to have integrated systems, to be terminally configured, and to be equipped with fly-by-wire flight control systems, all of which require highly reliable, fault-tolerant computers. Two candidate computer architectures are identified as having the potential of satisfying the commercial transport aircraft requirements:

- Software-Implemented Fault-Tolerance (SIFT) by Stanford Research Institute (SRI)
- Parallel-Hybrid Redundant Multiprocessor by Charles Stark Draper Laboratory (CSDL).

The system context of such computers indicates the need for some central complex that assures system collaboration and survival, which is the intended use of the SRI and CSDL concepts. The most stringent requirement imposed on these computers is that the probability of failure should be less than  $10^{-9}$  for a flight duration of ten hours. Descriptions of the SRI and CSDL concepts are presented and the most critical design issues are discussed. Reliability estimates are presented for the SRI and CSDL concepts showing their ability to meet this stringent requirement.

INTRODUCTION

NASA, in the past few years, has considered the application of advanced concepts of aerodynamics, terminal area control, and flight control [1, 2, 3] to commercial transport aircraft. Many of these concepts have shown a need for sophisticated avionic and control systems employing computers with some of the applications having safety-critical requirements. An initial study [4, 5] was conducted to consider the design of computer architectures to include in particular:

- The computational and reliability requirements of an advanced transonic commercial transport aircraft using fly-by-wire techniques with a unified digital computing system
- The impact of modern digital circuit technology on the design of such a computer
- The identification of candidate architectures for a computer to satisfy the requirements.

From this effort, two candidate architectures were identified as having the potential of satisfying the commercial transport aircraft requirements:

- A newly conceived architectural concept, Software Implemented Fault Tolerance (SIFT) by Stanford Research Institute (SRI)
- An existing architectural concept, a Parallel-Hybrid Redundant Multiprocessor by Charles Stark Draper Laboratory (CSDL).

Additional efforts are being conducted whose objectives are:

- To develop the SRI and CSDL design concepts to a point at which their potential reliability may be evaluated with reasonable accuracy
- To investigate alternate strategies for physical implementation using available or specially designed components
- To prove or substantiate the correctness of the hardware and software designs
- To model the systems and evaluate their effectiveness in tolerating faults.

To achieve the above goals, the current research was directed at the critical aspects of the designs, leaving less critical aspects to a later phase in the research program. Since this is an ongoing research investigation, the design and specification of the SRI and CSDL systems are not complete.

This paper discusses the context of the system within which the computer functions, the most stringent requirement imposed on such computers, provides a description of the SRI and CSDL computer concepts and discusses their ability to meet the most stringent requirements.

#### SYSTEM CONTEXT

For contemporary transport aircraft systems, redundancy occurs in the form of independent and often dissimilar subsystems, as shown in Figure 1. Subsystem failure is a routine occurrence and in most cases does not have a catastrophic impact on flight safety. To implement systems in this manner, the flight crew is employed as a system integrator with ultimate responsibility for failure detection, identification and recovery for the vehicle.

The implementation of substantially more sophisticated and autonomous controls anticipated for future transport aircraft calls for capabilities that require the creation of an integrated system. It is appropriate to regard the aircraft avionics and controls as a system of sensors and effectors where information processing elements derive inputs from sensors and generate control signals to effectors. Effectors, which are displays and actuators, operate upon the man-machine environment via dynamics and human responses thus producing effects that are measured by sensors.

With the future incorporation of such life-critical functions as active controls, total fly-by-wire, and system management, the system embraces both non-critical subsystems, critical subsystems and critical self survival of the system. A significant aspect of the system architecture, therefore, involves the assignment of redundancy and survivability to the portions of the system. Survival of the aircraft will require the survival of minimum levels of sensors and effectors, motive power, structural integrity, and the information processing system. Survival of the information system will require not only the survival of minimum levels of processing for sensors and effectors but also will further require the survival of system integrity, which is the unambiguous successful collaboration of surviving modules. Therefore, the survival of the aircraft depends not only on the survival of a suitable minimum subset of the sensors and effectors but also on the ability of the system to "mobilize" them into coordinated action. Such mobilization requires redundancy management wherein the vehicle dynamics are controlled via surviving effectors on the basis of information derived from surviving sensors.

Since the occurrence of unlikely events can never be ruled out, the design of such a system must resort to probabilistic criteria. Without going into detail, we can identify broad characterizations such as Mean Time Between Failure (MTBF), failure independence (i.e., absence of correlation among failures in distinct subsystems), and robustness in the sense of recovering from all failures for which it might potentially be possible to recover. The last of these items cannot be directly quantified. It requires the anticipation of all permutations of failures that will occur in the lifetime of a fleet of systems and the creation of "contingent modes" of operation.

The desirable characteristics mentioned in the preceding paragraph for systems and subsystems suggest the following design guidelines for critical integrated systems. First, sensors and effectors should be diversified to the point where failure correlations are adequately small. Diversification should not be excessive, however, as it is counterproductive to the second guideline, modularity. Modularity means the use of identical modules for multiple functions as well as redundancy and serves to fulfill MTBF, logistics, and contingency requirements. Modularity is beneficial as it tends to minimize the variety of contingency modes.

The highly survivable system could in principle be located in a central complex with dedicated connections to each sensor and effector. As a practical matter, a certain degree of distribution of the system is inevitable. Most of the sensors in use today, as well as those envisioned for the future, rely on electronics for manipulation, interpretation, and testing. A good example of a future effector is the development of more fuel efficient engines through the use of computers as engine controllers. In times past, the notion of system integration carried the implication of using a single large computer on a shared basis for as much of the processing as possible. Rather than trying to push information processing towards a central facility, the tendency today is to take advantage of the potential for distribution. This means not only the employment of specialized electronics for each subsystem but also the use of an appropriate amount of digital computer processing local to and/or dedicated to the subsystem.

System integrity can in principle be embodied in a wholly distributed system to some degree; however, incorporation of adequate redundancy and contingency for critical self-survival without possible detrimental effects from the non-critical subsystems poses significant survival problems. Therefore, the subject system architecture would employ a fault-tolerant central computer for the maintenance of system integrity. The adoption of a fault-tolerant central computer into the system structure makes it possible to assign both the redundancy management and the contingencies in an unambiguous way to a hardware-software entity whose failure and nonavailability are highly improbable.

With this system approach, the computer also serves those digital computation functions that involve the coordination of separate subsystems which are both critical and non-critical. Between the central computer and the local processors is a data communication facility which can be composed of non-critical elements, but whose partial survival is critical. Representative system structures are shown in Figure 2. Three different fundamental fault-tolerant data communications structures can be distinguished. They are the dedicated or star connection of Figure 2a, the bus connection of Figure 2b, and the network connection of Figure 2c [6]. Each approach has pro and con characteristics which will not be treated here in detail.

It may be presumed that all three of these methods will employ serial data transmission and that the interface to each subsystem will be comparable for the different methods. The interface at the central computer's end is virtually identical for the three systems differing only in the number of input-output access channels needed. There is no reason why all three of these methods cannot be used in different parts of the same system.

Within the system context previously described, the fault-tolerant central computer can be characterized as having the primary function of simply surviving with secondary functions of system redundancy management and contingency management. In addition to these functions, the central computer has other natural system roles owing to its hierarchical position in the system structure. These roles include the coordination of sensor-effector activities of which a digital autopilot is an excellent example. In some cases this function alone can account for a substantial fraction of the computing resources. Another role is that of command where there may be interaction with the flight crew.

#### REQUIREMENTS

A number of studies [5, 7] have been conducted to determine the requirements placed upon a computer for use in future commercial transports. The purposes of these studies were to develop estimates of the computational requirements imposed by contemporary and projected future avionics and aircraft functions and also to identify the safety levels of the computations. These estimates are intended to be used for determining the computational size and power required of the digital computer, to specify other constraints such as reconfiguration time, and to aid in developing a specification for the reliability of the computer.

Table I is a summary of the present and projected future aircraft functions and their safety criticality levels by flight phase. The flight phases constitute the major operational modes for which a computer must provide task allocation and scheduling among the processors and memories. Not shown by Table I are the additional tasks required for system redundancy and contingency management, and computer self survival. The details of the tasks will not be presented here; however, there are several important characteristics of the aircraft function task set. First, it should be recognized that different aircraft will place different computational loads upon a computer. Second, the set of tasks to be computed varies in the amount of computation and in the speed in which it must be carried out. Third, all of the computational tasks are repetitive in nature. Fourth, the tasks that must be carried out fastest are tasks with small programs and data sets. Fifth, for a typical task set the amount of data flow from one task to another is low.

Typically, a central computer might have placed upon it by the aircraft functions the following computational and sizing requirements:

- Processor Speed: 200 - 500 thousand instructions per second
- Memory Size: 16 - 24 thousand words.

From Table I, it is seen that the highest safety levels are imposed on projected future functions for stability and control. These safety levels are governed by the Federal Aviation Administration regulations for the design of flight control systems and other equipment, systems and installations in commercial transports, and the safety level is stated as "The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable". A number of less than or equal to  $1 \times 10^{-9}$  has been imposed in recent certification programs to represent an extremely improbable event. However, the application of the  $10^{-9}$  number to a system or subsystem is subject to interpretation and is usually examined on an individual basis by the FAA. For research purposes, the reliability requirement for the SRI and CSDL computers was established as the probability of system failure should be less than  $10^{-9}$  for the longest flight duration of ten hours. This is the most stringent interpretation that could be anticipated and places a high availability requirement on the computer.

The architecture of the central computer is strongly driven by the urgency of preserving a valid data stream. Failure to do so could have catastrophic consequences such as loss of the aircraft state vector, loss of configuration data, or loss of command. For this reason, data that is transferred, processed, or stored, is manipulated in replica. This data replication in combination with the task set requires a high degree of parallelism in the computer. Also, the potential task set is large and in the extreme case places a requirement for high computer productivity. Finally, to meet the different application possibilities, a requirement for expansion and/or contraction of the computer was established.

There is no unique parallel computer design. It can be a multiprocessor, a multicomputer, or a combination of the two. Additionally, numerous architectural alternatives exist. However, motivations for multiprocessors are typically to increase productivity and availability at the same time although these two purposes are largely competitive. Parallelism, productivity, availability, and expandability are intrinsic to the multiprocessor and through a previous study [4, 5], multiprocessors have shown the potential of satisfying the commercial transport aircraft requirements.

#### SOFTWARE-IMPLEMENTED FAULT-TOLERANCE

In recent years, a number of fault-tolerant architectures [8, 9, 10, 11] have been devised, analyzed, and in some cases, implemented. Most of these architectures depend heavily on special hardware structures to achieve their fault-tolerance. While hardware mechanisms are often useful for certain aspects of error detection and correction, they are limited in the kinds of faults they can treat. Also such mechanisms cannot be easily modified to reflect changes in performance and reliability requirements.



The SIFT (Software-Implemented Fault-Tolerance) computer [4, 12] is founded on a new approach to fault-tolerant computing that puts strong emphasis on software techniques for achieving reliability with corresponding de-emphasis on special hardware. The software that is critical to the reliability of the system is designed in accordance with a hierarchical design methodology [13, 14, 15] that allows one to state and prove properties relating to the system's correctness. A Markov process model is used to analyze SIFT's reliability as a function of various error-detection and reconfiguration strategies. The reliability model is incorporated into SIFT's formal description permitting one to show that the model indeed reflects the behavior of the system.

#### The SIFT Design

This section will describe the SIFT computer design, the manner of its operation and also the way in which the principal parameters of the design were derived from the requirements and the technological factors.

The major units of the computer are shown in Figure 3. The central processing units and associated memories are indicated by P and M respectively. The Input/Output processors and their associated memories are indicated by P<sub>io</sub> and M<sub>io</sub>. The P and M units form a computing module with the connection between them in a conventional manner by a high bandwidth bus. This connection enables the processor to obtain data and instructions and to place data and results in the memory unit. The computing modules are connected to each other by a system of several busses (in Figure 3, three such busses are shown, but this number can be varied). These intermodule busses are unidirectional in that a module may only read data from another module and may not write into the memory of another module. This restriction prevents a faulty module from having an adverse effect on operative modules thereby providing fault isolation between modules. The input/output subsystems (i.e., the P<sub>io</sub> and M<sub>io</sub> units) is also connected to the intermodule busses and its operation is described in the following discussion.

Computational tasks are placed in modules with tasks of high criticality being replicated in several modules while less critical tasks are placed in a lesser number of modules. This is illustrated in Figure 4 within each module the tasks' are multiprogrammed according to a fixed schedule. No attempt is made to carry out tight synchronization of the modules. It is merely necessary that the synchronization is sufficient for the timing constraints of the application and for the error detection and correction strategies that are used.

Within each module there is a local executive that controls all the operations within that module. The local executive includes scheduling, dispatching, error detection, error correction and error reporting. The coordination of all the modules is carried out by a Global Executive that is placed in only some of the modules. This global executive is responsible for reconfiguration and the associated diagnosis and check-out functions that must be carried out in the event of a fault, or when there must be a change in the task set to be computed, such as occurs during change of flight phase of the aircraft. This global executive is replicated in a sufficient number of modules to achieve the necessary reliability of the system as a whole.

The basic scheme for error detection and correction is to carry out each calculation in a number of modules and for the processor of each module to place the results in its own memory. The results are validated at the time that they are used rather than when they are computed. Any task that uses the result of some task (including possibly the next iteration of that same task) reads the several versions of the data it needs and carries out a comparison of the several versions. If all versions are the same, then the presumption is that no error has occurred and the calculation proceeds. This sequence of operations is illustrated in Figure 5. If one of the versions of the data is found to be different then it is presumed that it is in error. In this event, the majority vote of the versions of the data is used and the calculation proceeds after taking note of the fact that a data discrepancy exists. This note like any other data is written into the memory of the module that detected the discrepancy. At some later but short time, the next repetition of the global executive will read the notes left by the other modules that report errors that they have detected. The global executive then initiates a diagnostic program that identifies the unit that is faulty based upon the error reports that the global executive has read from all the modules.

In the event that this diagnosis perceives that a processing module is faulty, reconfiguration takes place to remove that unit from further effect on the system. The reconfiguration takes place according to the following scheme.

1. The global executive receives error reports from the several local executives.
2. A diagnostic routine is started which identifies the processing unit that is at fault.
3. The global executive determines those changes in task schedules that must be carried out to provide adequate redundancy for the task set. These changes are noted in tables that are in the memories of the processing modules that are computing the global executives.
4. The local executives in each of the modules (including possibly the faulty module) read the tables of schedules from the several versions of the global executives and on the basis of these tables change the task set on which they are computing.

The global executives referred to above are replicated versions of the same program and as such they should all produce the same tables of schedules. In the event that one of the global executives is running in the processing module that is at fault, then the possible erroneous results from that version will be ignored by the voting process that takes place whenever a module reads data from any other module. Thus, the reliability mechanisms that are used for the global executives are the same as those used for all other task programs and the global executive does not form a special "hard core" that needs any special or different treatment to achieve reliable operation.



The inter-module busses are used for communication between modules. When data has to be read from other modules, the bus used for each version of the data will be chosen so that a different bus is used for each version of the data. One faulty bus will therefore not produce erroneous results in the system because the voting process in each module will over-ride the non-agreeing data item. The error reports that are left by a module for the global executive to read will contain notes about the busses that were used for data transfer of the non-agreeing data item. On the basis of these data, the diagnostic program in the global executive can decide that it is a bus that is faulty rather than a processing module. The faulty bus is removed from further use by changing the tables that designate which busses are to be used. These tables in the global executive are read by the local executives which then change their own bus utilization tables thereby ceasing to use the faulty bus.

Many fault-tolerant computer designs that use replication to achieve fault-detection and/or fault-correction demand that the various versions of a computation of a task be carried out at identically the same time (so called "lock-step" operation). This implies that a system-wide clock of high reliability be included. In SIFT, this is avoided by only demanding a loose form of synchronization derived from the timing constraints of the task set. The basic synchronization rule is that no task may commence an iteration of its computation before all the other replications of that task have completed the previous iteration. This means that at any time the results of a computation are always available in the system for some recent iteration. In order for this scheme to work, it is necessary for each computation to use two buffers for the results. While one buffer contains the complete results of one iteration, the other buffer is being filled with the results of the next. The synchronization rule quoted above ensures that the first buffer will not be overwritten until the next iteration completes. The use of this loose form of synchronization has benefits in that a system-wide transient of short duration will be unlikely to affect all of the processing modules in the same way thereby preventing the occurrence of multiple correlated errors which cannot be detected by simple replication and voting.

The interconnection between the modules and the busses can be by a high impedance connection so that damage propagation from one type of unit to another can be achieved. It is also feasible to use an optical coupling between units as a means of preventing damage propagation. It must be shown in a fault-tolerant computer that a faulty unit cannot disrupt the system by any actions that it takes. The software voting that is used in the SIFT design provides such protection for all computational errors. However, it is also necessary to show that control actions by one unit cannot cause disruption of other units. All units of the system are constrained from forcing other units to carry out an action. They may only request action from other units, and the other unit protects itself by only acting on requests that it can carry out without disruption of its own operation. An example of this is the fact that a module may only read from another module's memory and may not write into it. One form of disruption would occur if a faulty unit made excessive requests upon another unit thereby preventing it from carrying out other actions including the serving of requests from yet other units. This is prevented by logic in each unit that scans all units that request service and only honoring this request for a small amount of time before continuing to scan the other units. This mechanism exists in two places. First, a bus will only serve a requesting processor for a one-word transfer before checking whether other processors also require service. Second, a memory unit within a module will only transfer one word to a bus before serving other busses that may request service. This mechanism ensures that a unit that becomes faulty cannot prevent other units from serving other requests. As this request for data is the only control function that spans more than one unit, the design provides fault isolation across units.

The input/output subsystem is connected to the bus system as shown in Figure 3. The fault-tolerance techniques that are used are as follows:

- Critical sensors are replicated, and the programs that require the data read all the versions and carry out a voting procedure.
- Critical actuators are replicated and each of them contains sufficient local logic that they can read the several versions of the output data that they require and carry out local voting possibly by mechanisms similar to those currently employed with multiple actuators on aircraft (e.g., forced sum voting).
- Non-critical sensors and actuators are not replicated but are connected to SIFT in the same way as critical ones in order to preserve the fault-isolation rules on the input/output subsystem as are used between processing modules.

For critical input and output units, the data to and from the SIFT system flows on a multiple bus system which is connected to the main bus system of SIFT via logic that is realized by a specially programmed micro-processor ( $P_{IO}$  in Figure 3).

Each microprocessor operates in a similar manner to the main processors of SIFT except that the tasks that are to be performed are much smaller and the executive that resides in them is a reduced version of the executives in the central processors. The reductions that are made are as follows:

- No global executive is present in the microprocessors, as the functions normally performed by it are either not necessary or are carried out by the global executive of the central processors.
- The local executive contains only the voter, scheduling, and dispatching functions that enable it to determine its schedules by reading the central global executive tables.

In all other respects the I/O processors operate according to the same general rules as the central processors. These operations include voting on multiple inputs to achieve error detection and correction, reconfiguration by change of scheduling tables, and the restriction that a processor may only read data from other processors and may not write into the memory of other processors.

In a SIFT system that is carrying out both critical and non-critical tasks, it is necessary to maintain a separation between the tasks because the non-critical tasks may not receive as much validation and verification as the critical tasks and thus may corrupt them. This is achieved by the use of different  $P_{i0}$  units that connect inputs and output units to the main bus system. The  $P_{i0}$  units are used to read from sensors and deposit the results of the read operation in their own memories. These results can then be read by the main processors of SIFT. This scheme effectively isolates potentially unreliable sensors from the other units of the system. Non-critical output units are handled by an analogous scheme in which  $P_{i0}$  units read from the main memories and transmit data to those actuators.

#### The Design Methodology

The SIFT design has been specified in accordance with a formal design methodology that originated with D. Parnas [16, 17] and has been extensively developed at SRI (Robinson et al. [13]). The chief reasons for using such a medium were (1) to impose a discipline on the design process assuring a clearly-structured, easily modified design, (2) to simplify verification of the correctness of that design, and (3) to facilitate the analysis of certain reliability properties. Previous use of the methodology has been concerned with only the first two of these aims. The SIFT effort is the first instance of its use in connection with reliability modeling.

The methodology can be viewed as a formalization of Dijkstra's stepwise refinement concept [18]. The central idea is that of decomposing the design into a hierarchy of abstract virtual machines or Parnas modules. The highest modules in the hierarchy provide an abstract, global description of the system's capabilities. Modules at lower levels of the hierarchy serve as building blocks for implementing the highest-level module. Modules at still lower levels are building blocks for implementing those at intermediate levels. The modules lying near the top of the hierarchy thus tend to be highly abstract while those at or near the bottom tend to be more concrete. In the SIFT design, for example, descriptions of real machine hardware appear at the bottom level, and a set-theoretic model of the workings of the system appears near the top.

Each module in the hierarchy is specified in terms of a set of abstract data structures and a set of operations that change the values of these structures. At any given moment, the state of the module is determined by the aggregate of the values of its data structures. Operation calls thus cause transitions from one state to another. The data structures and operations of each module are specified in a formal way. The specifications describe what happens when each of the functions of a module are called. Specifications for operations consist of assertions, such as logical formulas that relate the state of the module before an operation call, to the state resulting from the call. Module specifications have other aspects [19] that are not directly relevant to this discussion.

Each module (other than those at the very bottom of the hierarchy) is abstractly implemented in terms of those modules lying immediately beneath it in the hierarchical ordering. First, a mapping function is specified that maps states of the implementing modules to a state of the implemented module. The mapping function thus represents the data structure of the implemented module in terms of those of the implementing modules. The selection of mapping functions for an implemented module corresponds to deciding on the data structures for that module. Next, for each operation of the implemented module, an abstract program is specified. The abstract program is expressed in terms of the data structure and operations of the implementing modules and is intended to mimic calls of the implemented function as a sequence of calls to operations in the lower-level modules.

Two types of proof can be carried out according to the methodology. First, the modules that constitute the top level in the hierarchy contain the interface functions, i.e., the functions that the system provides to users. Based only on the specifications of these modules certain properties of the system can be deduced. Second, the implementation of the modules can be proven correct level-by-level. Each abstract program is proven correct with respect to the specifications of the operations it implements and with respect to the appropriate mapping function. Using the methodology, the proof of a large system can be reduced to that of many small programs.

#### Structure of SIFT Software

The logical structure of SIFT is depicted in Figure 6. It consists of a hierarchical layering of modules, which we designate as system modules, and some programs--namely the application tasks and the global and local executives--that utilize the facilities of the external interface of the system modules. For simplicity we will say that the tasks call the functions of the interface. Each of the system modules may be considered as an abstract machine that maintains a state and provides operations to modify the state. The application tasks and executive may then be considered as programs that run on the abstract machines. The data required by the tasks is distributed among the system modules.

The structure shown in the Figure 6, with a few exceptions, appears in each of the processors. Each task, including the global executive executes in some subset of the processors. Thus the fault status and fault schedules modules, which are accessed only by the global executive, appear only in processors executing the global executive. Naturally, the state of a module at a given instance varies according to the processor with which it is associated.

Figure 6 depicts some additional inputs--clock-tick, timer and faults. These are operations of particular modules that can be viewed as being "called" by processes that are external to the system and operate synchronously with the processing of tasks.

#### Reliability Model of SIFT

In our discussion of the design methodology, it was noted that the top-level modules provide a complete external description of the capabilities of the system. Properties asserting the correctness of the system are stated and proved relative to the specification of these modules alone. The other

modules in the design have no other purpose than to facilitate the implementation of the top-level module. The proof of correctness of the implementation is important only in that it guarantees that the specifications of the top-level module are satisfied.

These specifications, of course, relate only to the functional behavior of the system--not to its reliability properties. The reliability model, on the other hand, describes the probability of certain failures without any particular reference to the functioning of the system in the event of such failures. Clearly, the meaningfulness of the reliability model depends heavily on this behavior. In order to have any confidence in the applicability of the model, it is necessary to demonstrate that its various states do in fact correspond to the appropriate states of the top-level module of the design. In particular, it must be proved that the sequence of events that lead to the Fail State in the reliability model exactly correspond to sequences of events that lead to failure in the top-level module of the system.

To facilitate this proof, the reliability model is formulated as an integral component of the hierarchical design. A new module, called the reliability module, is positioned above the former top-level module replacing that module as the most abstract description of the system. The new module's only data structure encodes the state of the reliability model and its operation models failure events and reconfigurations.

In the use of the SIFT system for aircraft control, the significant states of the system will depend on changes of flight phase and on any faults that have occurred. The dynamic nature of the fault-tolerance techniques that are used (e.g., varying task replication at varying times) make it important to analyze these different system states. For each state of the system, the probability of transition to other states is a function of the variables that describe the state presently occupied, e.g., number of remaining good processors. This model of the system behavior is particularly attractive because of its close parallel to the operation of the real system and also because of its mathematical tractability.

Calculations based on this reliability model show (Figure 7) that a SIFT system with five processors and four busses initially would have less than  $10^{-9}$  probability of failing to have sufficient computing resources at end of a ten-hour flight. This meets the required reliability objectives as previously discussed.

#### Concluding Remarks

In the SIFT design, the major parameters are derived from the requirements and from the opportunities that are presented by recent technological advances.

The use of software for error detection and correction is possible because the loose connection between tasks implies that the amount of data that must be moved from task to task is small. Software reconfiguration is also possible because the time to accomplish it is acceptable when viewed from the time constraints on the tasks and the fact that the fastest tasks tend to be small. This allows for the movement of complete tasks from one module to another.

The low data transfer that is necessary between tasks allows for a bus structure that is slow thereby allowing the use of mechanism that achieve fault isolation between units.

The low cost of modern electronics allows the policy of reconfiguring on the basis of complete computing modules or busses. Many previous fault-tolerant computer designs carried out such reconfiguration on the basis of much smaller units. Such designs tend to be very complex and are currently unjustified in view of the low costs of modern electronics.

In addition to the fact that the design is driven by the requirements and technological advances, there are many other advantages that stem from the use of software as the principal techniques for achieving fault tolerance. These include:

- The degree of fault tolerance can be different for different tasks within the task set.
- The degree of fault tolerance can be different for the same task at different times, e.g., during different flight phases.
- The total computational power available to the tasks can be varied by changing the number of computing modules.
- There are no special fault-tolerance restrictions on the processing modules and therefore standard off-the-shelf units can be used with the benefit that they will have experienced far more thorough validation and verification than specially designed units.

The SIFT concept embodies a number of ideas whose usefulness extends beyond the particular application for which the system was intended. Because conventional, off-the-shelf processing units comprise the bulk of the hardware, the core system can be easily and inexpensively adapted to a broad range of needs. However, because the degree of reliability achieved by the system depends on the number of processors used and on scheduling strategies rather than on built-in aspects of the design, it can be varied according to the performance and cost requirements of the application.

The use of a formal design medium for purposes of specification, validation, and reliability modeling can be expected to play an important role in future designs of fault-tolerant computers. While a system might make extensive use of redundancy, unless the software or hardware mechanisms that manage the redundancy are correct, the system will still be unreliable. Similarly, the formulation and use of elaborate reliability models is to little avail if it can not be assured that these models actually reflect the behavior of the system. We believe that SIFT constitutes a major step in the direction of fault-tolerant systems whose correctness and reliability can be verified.



A PARALLEL-HYBRID REDUNDANT MULTIPROCESSOR

The physical organization of the parallel-hybrid redundant multiprocessor is substantially more complex than a nominal multiprocessor organization. A simplified module diagram of the computer is shown in Figure 8. Superficially, this diagram appears the same as a nominal multiprocessor. The principal differences are that the busses for memory and interface access are redundant and that the actual number of modules is three times the number of nominal modules plus some number of spares.

All activity is conducted by triads of modules and triads of busses. A module triad is formed by associating any three like modules with one another. This means that any module can serve as a spare for any triad. Such flexibility permits the best possible utilization of surviving modules. A single triad of bus lines is active at any one time for each of the memory and interface accesses. In other words, a three-member subset of N bus lines is chosen on a quasi-static basis to serve as a bus triad.

Every module of every kind is able to receive data from all incident bus lines and contains a decision element to formulate a corrected version of bus data. It is necessary for each module to know which three bus lines are the active ones. These three lines are connected to a voter in each module, thus constituting a Triple Modular Redundant (TMR) element. The three active bus lines carry three independently-generated versions of the data with each version coming from a different member of the triad that is transmitting the data. To accomplish this, it is necessary to assign each module to transmit on one specific bus line. Now if totally flexible module configuration is to be possible, it follows that the assignment of a module's transmission to a single bus line must be quasi-static and reconfigurable.

In addition to the redundancy described in the preceding few paragraphs, the redundant organization differs from the nominal one by virtue of the inclusion of independent submodules called bus guardians (BG) units in each processor, memory, and input-output access unit. Guardians are charged with governing the status of their associated modules. This includes power-on status, memory bus triad and transmission selection, and certain self-test configuration selections.

Each of the functions of the guardian has the characteristic that its failure modes have safe directions as well as unsafe ones. By biasing the failure modes toward the safe directions, it is possible to increase the probability of system survival. In general, the safe failure modes of a module are power-off and bus transmission disconnected. To bias in this direction, one can employ redundant guardians in each module, and require agreement among them to establish power-on and bus transmission enable.

The connection of bus guardians is illustrated in Figure 9. It should first be noted that the guardian principle depends heavily on fault independence. Therefore each guardian derives its power, its bus inputs, and its timing reference independently of all other guardians. It is moreover physically isolated from all other guardians and all modules. A particularly critical area from the isolation viewpoint is the control of the module's transmission interface onto the various bus lines. The bus isolation gates (BIGS) must be highly independent of one another as must the guardian's enable signals to these gates. This is one of the crucial electrical and mechanical design aspects of the entire computer.

Bus guardians are addressable as part of the common memory address space and are capable of receiving messages from any processor triad via the active memory bus triad. A message to a guardian contains commands which are staticized by the guardian and applied to its outputs until superseded by a new command message. In this way, the probability is remote that a failed module can assert more than one erroneous data stream. As a result, correct data can be determined by the bus voters, and the malfunctioning module can be switched to a silent state. It is noted in passing that certain failures of a bus isolation gate can render a bus line useless, in which case that active bus triad must be reconfigured to use a spare line. However, most guardian failures appear as passive failures of the processor, memory, or input-output access unit to which the particular guardian unit pertains. Guardians are used as agents to convey the computer's configuration authority to all elements of the computer. They are highly secure against the random or willful malfunction of any single active transmitting module. They make possible highly flexible reconfiguration.

All modules and buses are organized into triads. In the case of processors and memories, there can be numerous triads in existence at the same time, but only one memory bus triad and only one interface bus triad. Each processor triad acts as one functional processor, of which several can work in parallel. Each memory triad acts as a page of memory, of which several can exist at one time, but only one can communicate at a time with a processor triad.

When a processor fails, its triad will attempt to complete its current job step which it will be able to do unless a second failure prevents it. The period of vulnerability to a second failure will be a fraction of a second. When the job step is complete, one of the processor triads is assigned the task of reconfiguring the injured triad. When the erroneous module is identified, it is removed by commands to its guardians. If a spare is available, it is connected to the appropriate bus by its guardians, likewise upon command by the processor triad assigned to the reconfiguration. Triad identity will be assigned to the spare processor by a direct message. If no spares are available, the injured triad is retired. The resources of the multiprocessor are diminished by one processing unit, and the two unfailed members of the former triad are now available to be used as spares if further failures occur.

The situation is much the same for memory modules. The principal difference is that memories are not anonymous. In fact, a read-only memory module is totally dedicated to its assigned function and cannot be used as a spare. When a read-only memory triad is injured by the loss of a memory module, a read-write memory module can be used as a spare. It must be loaded to agree with the surviving triad members before a second failure occurs. If no spare is available, the triad is reduced to a dyad which is vulnerable to the next failure, at which time one memory page is lost. This is a significant



departure from the flexibility offered by the anonymous processor triads. The eventuality of read-only memory failure must clearly be covered by the inclusion of adequate spares either read-write memories for flexible pooled use or extra dedicated copies of read-only memory.

Figure 8 indicates the existence of input-output access modules connected to the internal interface bus and also to the external environment. It should be pointed out that the external interfaces of the computer could alternatively support dedicated, bussed, or networked link structures to the sensor and effector subsystems. The redundancy structure at this point depends on the redundancy desired in the external interface.

The simplest conceptual structure is for a triple-redundant interface, such as a redundant external bus, where the triple module redundancy structure is extended through to the subsystem interfaces. Each external bus line can be dedicated to a different input-output access module, which in turn is assigned by its guardian units to transmit on one of the active interface bus lines. More complex variants are possible in which each access module performs error correction by voting on incoming data from the external bus.

When an external interface is non-redundant, the strategy would be to assign it to a single access module where the module would transmit on all three active interface bus lines. A malfunctioning access module could pollute the entire interface bus, but with suitable encoding and protocol there would be no serious consequences to the state of the system. The offending access module could be discovered and disconnected by bus guardian commands conducted over the memory bus, the major penalty being a time loss on the remainder of the input-output interface of the computer. For dedicated links, the loss of the link is non-critical by hypothesis. For a network, whose survival is assumed critical, the computer must interface with the network in several places via several distinct access modules. Each such interface would be simplex, but the system would survive the failure of all but one of them.

The employment of independent redundancy requires some form of synchronization among the independent data sources. Soft, or loose synchronization involves such operations as buffering, comparing or voting, signalling consensus, and marking completed intervals. These can all be done by program when given suitable intermodule data links. Hard or tight synchronization involves hardware comparison or voting and a common time reference where loose synchronization can employ separate time references.

Tight synchronization is employed in the parallel hybrid redundant multiprocessor. It provides the basis for solving some problems and presents some problems of its own. A common time reference, or clock, that supports hardware voting allows instantaneous validation of internal data, configuration control, and, in some cases, interface data. In this way, it helps to make the redundant multiprocessor resemble the nominal one which is advantageous to programmers at all levels.

The fault-tolerant clock [10] shown in Figure 10 consists of a set of independent phase-locked oscillators arranged so that the failure of one or more of the oscillators (up to a design limit) does not destroy the phase lock of the survivors. The clock signal from each oscillator is distributed to every module and guardian so that each can make an independent determination of clocking edges. These independent determinations are made by circuits called clock receivers. In normal, nonfailed operation, the outputs of all the clock receivers are in phase lock with each other and with all the oscillators. The same phase lock holds when an oscillator fails. The failure of a clock distribution line appears as an oscillator failure, and the failure of a clock receiver appears as a failure of the module or guardian that contains it.

#### Fault Detection, Identification and Recovery

The central computer is designed to have a highly improbable loss of capability. One can roughly quantify this statement by saying that one of these computers should exhibit a total failure rate of less than  $10^{-9}$  in a flight of up to ten hours. This virtually rules out the use of ordinary triple modular redundancy, as the MTBF's achievable in large scale production have been consistently too low for such reliability without replacement of failed modules. Therefore some form of hybrid redundancy is needed. In a simplistic view, hybrid redundancy works by substituting a spare the first time the TMR voters disagree. This view has the shortcoming of not taking latency of faults into account. That is, the first fault may not result in any voter disagreements; whereas when combined with a second fault, it may frustrate recovery. A pre-requisite for achieving highly improbable failure in a hybrid system is to expose latent faults by systematic exercising, or "flexing" of all logic elements. The question remains of how often such flexing must occur. Hopkins and Smith [21] have shown that the flexing period must be of the order of seconds for a reasonably sized system with module MTBF's in the ten-thousand hour range. Clearly, then, flexing cannot be relegated to pre-flight checkout, but must rather be conducted routinely in flight. An ordinary hybrid TMR system cannot routinely test itself when performing critical functions as it is vulnerable during these times. A parallel hybrid TMR system can do this becoming an integral part of the computer's architecture.

The latency problem poses an interesting design dilemma. Redundancy is employed to mask the effects of faults upon the system as a whole. But redundancy requires flexing of all logic and requires that all possible faults that are created by flexing be made visible to the system and not masked. The resolution of this dilemma requires reconfiguration of all independent system elements plus the selective generation of faulty symptoms to verify detection mechanisms. This is why, for example, quadded or interwoven [22] logic is not proposed for this application as it cannot be reconfigured and tested on line. The same holds for many of the error-correcting coded memory and arithmetic units that have been designed.

In the parallel-hybrid redundant multiprocessor, an error correction mechanism exists in every module in the form of a voter. Each voter must be tested routinely to ensure that its error correcting capability is undiminished. Of all the voters, only those in the processor modules have the additional capacity to detect as well as to correct errors. This is not to save equipment. Rather, the processor

is the only kind of module in the computer that can utilize the information. Processor bus voters under normal conditions will correct single bus errors and will set error latches to indicate which of the buses was in disagreement. At this time, the processor can record the identity of the nominal user of the bus for diagnostic purposes. A processor triad can flex its own voters during a test job step by having each triad member purposely utter independent bus data that causes all possible kinds of bus errors. To pass the test, all triad members must receive the same data, form the same corrected result, and indicate the same disagreement patterns in their error latches. This is a relatively simple test procedure which can be conducted by a processor triad under test while other triads carry on normal functions. In a sense it qualifies the triad to conduct further testing in which the triad's voters are the decision elements. The remainder of the system testing function is carried out under the assumption that the processor voters and error latches are operational. The test process involves the conversion of every fault into an error by making calculations whose results are sensitive to each logic variable. Each bus and module, including voters, guardians, isolation gates, clock receiver, oscillators, and data and power interfaces must be exercised in depth.

Processor testing involves fairly conventional self-test approaches except that coverage needs to be higher than that which is typically obtained in computers. A guideline, then, for processor design is to eliminate obscure and pattern-sensitive sequences as much as possible. The cache memory is also tested by a conventional program approach. Address faults and pattern sensitivities present the most important problems to be solved.

Memory module testing is similar to cache memory testing. The memory voters are tested by sending single-error messages to a memory triad over the memory bus and verifying correct responses from the triad members.

Input-output access modules are also tested by messages from processors. Where voters are used, they are tested in a manner similar to memory voters. Simplex access units are tested in conjunction with input-output links.

Guardians are tested by reconfiguration commands and their voters are tested the same as memory voters by erroneous commands.

The clocking system presents a unique testing problem because it is nearly separate from the data handling elements and because the testing of clocking circuitry is fundamentally different from the testing of other logic circuits which are testable once a valid clock exists. Latencies in the oscillators can occur in the phase-locking circuitry. This is probably the most vulnerable area for in-flight testing. If the a priori probability of failure in phase lock circuits is sufficiently low, it may be possible to perform these tests only a preflight time. Clock receiver latencies, however, can be tested in one module at a time with minimal system vulnerability.

We might summarize the fault detection process as the arrival of disagreement errors at the voters of a processor triad which has been stimulated by normal or test activity. The detection of a fault initiates the process of fault identification which is the discovery of the module, bus, or other isolated element in which the failure resides. During the testing process for latent faults, there is relatively little ambiguity in the determination of faulty modules. In normal operation, however, an error on the bus can come from a number of sources. The identification of the faulty module generally requires the "rounding up of suspects," that is, the listing of elements that transmit on the disagreeing bus. If a module fault is permanent, the module can be found by moving it to another bus. If the bus is faulty, reconfiguration will not move the error to another bus.

Intermittent faults are less easy to identify. When the source of an error eludes detection by disappearing, all of the suspect elements are assigned one demerit, and a reconfiguration is then made to distribute the suspects evenly on different buses. Subsequent error occurrences and reconfigurations will cause a preponderance of demerits to accumulate in the name of the faulty module.

The recovery process is one of assignment and initialization for modules, and voter and transmitter selection for buses. These are all accomplished by the bus guardian units upon receipt of commands from active triads executing system software. Recovery can take place even if single errors are present on the buses. In principle, therefore, an injured processor triad can reconfigure itself.

#### Tolerance Renewal

The primary advantage of hybrid redundancy over TMR is that injured triads are reconfigured back to a state where they can once again mask malfunctions. This is a process of tolerance renewal. In principle, the system failure rate is restored to its design value by the reconfiguration process. If reconfiguration were to fail, the system failure rate would increase possibly by many orders of magnitude.

In practice, there are several ways in which an injured triad can fail to be reconfigured. These include exhaustion of spare modules, malfunction of the reconfiguration mechanism, failure to detect the need to reconfigure, and perhaps the use of a defective spare module. We can characterize the process of tolerance renewal as the detection and location of any physical malfunction, the removal of vulnerability from the triad containing the malfunction, the replacement, by spares, of functions thus removed, and the initialization of the reconstituted triad. All mechanisms involved in this process are subject to malfunction, of course, and such malfunctions constitute injury to their triads and require that tolerance renewal be carried out.

The idealized renewal of tolerance, together with a sufficient complement of spares, has an interesting theoretical consequence if the hazard rate is constant. Figure 11 illustrates this concept. The system failure probability increases monotonically with time in the absence of tolerance renewal. At any arbitrary point in time, if the system is tested and found to be perfect, the reliability becomes

equal to one. If the test reveals an injury, then the system is reconfigured to a state of virtual perfection and the reliability is likewise equal to one. This paradoxical result depends on the absence of any deterioration of the renewal mechanism, however. The concept's utility is to suggest to the reader that the dynamics of redundancy management are all-important in maintaining the system reliability at a level that is unreachable by non-redundant elements.

The actual system behavior differs in several respects from this concept. First, the supply of spares is not inexhaustable. Second the tolerance renewal mechanism is subject to degradation. Finally, the ability to test the system is limited by its probabilistic nature. These three items will be briefly discussed in the paragraphs following.

The supply of spares is a degree of freedom available to the system designer. In the parallel-hybrid redundant multiprocessor, there can be arbitrary numbers of processors, memories, interface access modules, memory bus lines, interface bus lines, oscillators, and power converters. If the failure rates of these elements are known and if the minimum numbers of each needed for system survival are known, the probability of exhaustion of spares as a function of time can be calculated using conventional combinatorial analysis.

The tolerance renewal mechanism in the parallel-hybrid redundant multi-processor is largely contained in the voters and the bus guardian units. Both the voters and the guardian units possess the bus line interfaces, and therefore are both capable of degrading elements (i.e., bus lines) outside of their own modules (e.g., processor, memory, interface access). This by itself is not qualitatively different from a single malfunction. The important concern is that all guardians in a single module may fail in such a way as to enable that module to transmit on more than one bus line. As mentioned previously, design steps are taken to minimize the probability of this eventuality, but the probability is finite that it will happen. A subsequent failure of the module in a malevolent state could cause an entire central computer to malfunction.

Finally, we deal with the problem of detecting malfunctions when they occur. If a malfunction results in a bus data malfunction, it can be detected by the voters. If not, it is termed "latent." Latent malfunctions must be considered at least as harmful as visible ones. It is for this reason that each critical element must be subjected to periodic test while the system is on line. As an approximation to the testing process we assume that the detection of malfunctions is an exponentially distributed random process with a constant rate of discovery. Figure 12 illustrates the nature of the distribution. According to this model, of all the possible malfunctions that will occur, most of them will be detected after a lapse of several time constants, but some will never be detected. The parameter  $\rho$  represents the rate of discovery. The system malfunction probability that results from finite malfunction detection time can be modeled as a Markov process with constant hazard rates and constant discovery rates.

The total probability of system failure is the composite of the parts previously discussed and is given in Figure 13 which illustrates the ability of the multiprocessor to meet the reliability requirement.

#### Reliability and Maintenance

The multiprocessor's design approach to system reliability consists of a combination of shielding, environmental control, redundancy, reconfiguration, test algorithms, voting, and high reliability design and manufacture of all hardware elements. In addition, the system software must be virtually perfect. To a certain extent, these facets of system reliability are synergistic. That is, once the central computer attains a certain degree of reliability, it becomes competent to serve as its own manager and thereby becomes capable of attaining even greater survivability by judicious utilization of resources.

Prior to each flight, the multiprocessor will test itself and the rest of the information system to purge it of latent malfunctions and establish accurately the degree of tolerance remaining. If this is adequate for dispatch, the flight will proceed maintaining high reliability by the tolerance renewal procedure including frequent testing of every element. In this way, a log is maintained of the status of every element of the system. Intermittent, transient, and permanent malfunctions can be distinguished, and the momentary tolerance made known to the flight crew. If changes in flight plan or envelope are called for, these can likewise be made known.

Upon landing, the need for maintenance, if any, of the information system will be readily discernable including in most cases the identities of the elements that have malfunctioned. A possible byproduct of system fault tolerance is the realization of considerable operational cost saving by postponing maintenance until the aircraft arrives at a base where this is most economically accomplished. If the probability of needing a module change earlier than one hundred flight hours can be held below one per cent, it could be well worth the inclusion of one or two extra spares to make this possible.

#### Multiprocessor Software Issues

The major system software elements are the executive, test, diagnostic, and system configuration programs, and the macro interpretation facilities. These elements are all closely related to the computer's architecture and are responsible for expediting job steps and for tolerance renewal.

The executive program embraces several sub-functions which are the time queue, the event queue, job dispatch, and the cache memory functions of invocation and retirement. The time queue and event queue programs maintain data files containing the identities of job steps that are scheduled to be executed. They also respectively contain for each job step the time or the enabling event that will signal the go-ahead for emplacement. They identify any necessary restrictions on emplacement such as insisting on or excluding a particular triad. Finally, a priority measure is maintained for each job step. In order to execute sampled-data control algorithms, it is necessary and sufficient to dispatch



the same job step at regular intervals. The executive automatically handles the iterative scheduling required by these algorithms. The job dispatch program is invoked by a processor triad as soon as possible after the triad has completed its execution of a job step. This program finds any non-excluded job steps that have become ready for emplacement, chooses the one whose priority is highest, and initiates an invocation of this job step. This triad then performs this job step, and the job dispatch program is released for use by other triads. If no job step is ready, the job dispatch program becomes reinvoked after a short delay to allow other triads to maintain normal resource access.

Cache memory management consists of the invocation and retirement of job steps and of the procedures and sub-procedures that constitute the job step activity. Invocation occurs for program portions that reside in, or are transferred to, the cache memory. Retirement occurs when such a portion must be overlaid. There is a strong analogy between cache management and page swapping in a hierarchical memory. The purpose is to relieve the applications programmers of most of the mechanics of moving their programs and data into the cache memory.

The reconfiguration program for the multiprocessor controls module and bus activity by sending messages to the guardian units. It maintains records of the status of each element and algorithmically responds to contingency situations. Reconfiguration programs will also exist for the external parts of the system primarily the input-output bus or network.

When a malfunction is detected, the diagnostic program is invoked. This program attempts to locate the malfunction source by using any diagnostic data available from the processor triad that detected the malfunction and by reconfiguring the computer so as to cause the malfunction source to move. If the malfunction does not recur, the demerit strategy described earlier is carried out. Similar programs will exist for diagnosing malfunctions in the parts of the information system external to the computer.

Macro interpretation facilities, like test programs, are quite specific to the logical design of the processors. They will exist as a combined machine language and microcode facility and will serve not only the applications programs but also the system programs. Cache management and other executive functions will probably depend largely upon macro operations. The choice of macro operations and the relative emphasis on machine language and microcode will be influenced by the applications program requirements.

#### Experimental Multiprocessor

In June 1976, an experimental multiprocessor at the Draper Laboratory, employing parallel-hybrid redundancy and closely resembling the multiprocessor described here, was used as a digital autopilot in a simulated Boeing 707 aircraft. The autopilot functions performed were minimal but totally critical to flight. During the simulation exercises, malfunctions were injected in a variety of ways into the multiprocessor, which successfully recovered in every case. Although still far from a highly fault-tolerant computer, this equipment has demonstrated the validity of the basic concept described here in many hundreds of hours of operation.

#### CONCLUSION

This paper has described the principal characteristics of two reliable multiprocessors for use in a system designed to be highly tolerant of malfunctions. These computers play a fault-tolerant central computer role in a distributed digital system and at the same time will be responsible for maintaining the operational status of the system and itself.

An experimental model of one of the computers has been built and operated in an aircraft flight simulation that exhibits its ability to survive isolated random malfunctions.

Designs are proceeding on several fronts for a computer of this type that can operate dependably in a real environment. When this type computer is coupled with systematically redundant sensors and effectors, their local processors, and valid application software, it will be capable of performing critical control functions in an aircraft with a suitably remote probability of failure.

Finally, these designs, while intended to be capable of satisfying the very stringent reliability requirements of a modern aircraft control computer, are also appropriate to a wider class of applications where high reliability is required.

#### REFERENCES

1. Leonard, R. W.; and Wagner, R. D.: "Airframe Technology for Energy Efficient Transport Aircraft", Presented at SAE - 1976, Aerospace Engineering and Manufacturing Meeting, San Diego, Ca., November 1976.
2. Reeder, J. P.; Taylor, R. T.; and Walsh, T. M.: "New Design and Operating Techniques For Improved Terminal Area Computability." Presented at SAE - Air Transportation Meeting; Dallas, Texas; May 1974.
3. Hood, R. V.: "A Summary of the Application of Active Controls Technology in the ATT System Studies", Presented at the Symposium on Advanced Control Technology and Its Potential for Future Transport Aircraft Los Angeles, Ca.; July 1974.
4. Wensley, J. H., et. al., "Design of a Fault Tolerant Airborne Digital Computer", Volume I - Architecture, NASA CR-132252, Stanford Research Institute, Menlo Park, California (October 1973).



5. Ratner, R. S., et.al., "Design of a Fault Tolerant Airborne Digital Computer," Volume II - Computational Requirements and Technology, NASA CR-132253, Stanford Research Institute, Menlo Park, California (October 1973).
6. Smith, T. B., "A Damage and Fault-Tolerant Input/Output Network", IEEE Trans Computers, May 1975.
7. Bjarman, B. E., et.al., "Airborne Advanced Reconfigurable Computer System (ARCS)," NASA CR-145024, Boeing Commercial Airplane Co., Seattle, Washington (August, 1976).
8. Avizienis, A., et.al.: "The STAR (Self Testing and Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE Trans., Vol. C-20, No. 11, pp. 1312-1321 (November 1971).
9. Hopkins, A. L., Jr.: "A Fault-Tolerant Information Processing Concept for Space Vehicles," IEEE Trans., Vol. C-20, No. 11, pp. 1394-1403 (November 1971).
10. Maison, F. P.: "The MECRA: A Self Reconfigurable Computer for Highly Reliable Process," IEEE Trans., Vol. C-20, No. 11, pp. 1382-1388 (November 1971).
11. "Design of a Modular Digital Computer System," Phase I Report, NASA CR-123655, Hughes Aircraft Company, Fullerton, California (April 1972).
12. Wensley, J. H.: "SIFT-Software Implemented Fault Tolerance," Proceedings of the Fall Joint Computer Conference, Vol. 41, pp. 243-253 (AFIPS Press Montvale, New Jersey, 1972).
13. Robinson, L.; Levitt, K. N.; Neumann, P. G.; and Saxena, A. K., "A Formal Methodology for the Design of Operating System Software," in Current Trends in Processing Methodology, Vol. 1, R.T. Yeh (ed.) Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976).
14. Robinson, L.; Levitt, K. N.; "Proof Techniques for Hierarchically Structured Programs", Communications of the Association of Computing Machinery (March, 1977).
15. Spitzen, J. M.; Levitt, K. N.; and Robinson, L.: "An Example of Hierarchical Design and Proof," AD#AO21 574/9WC, Stanford Research Institute, Menlo Park, California (March, 1976).
16. Parnas, D. L.: "Information Distribution Aspects of Design Methodology," in Proc. IFIP Congress 1971, North-Holland Publishing Co., Amsterdam (1972).
17. Parnas, D. L.: "A Technique for Module Specification With Examples," Comm. ACM., 15, 5, pp. 330-336 (May 1972).
18. Dijkstra, E. W.: "Notes on Structured Programing" in Structured Programing, Hoare, C.A.R. (ed.) (Academic Press, New York, New York, 1972).
19. Robinson, L. "Specification Techniques," Proceedings of the Thirteenth Design Automation Conference (June 1976).
20. Daly, W.; Hopkins, A.; McKenna, J.: "A Fault-Tolerant Digital Clocking System," Digest 1973 International Symposium on Fault-Tolerant Computing.
21. Hopkins, A. L., and Smith, T. B., "The Architectural Elements of a Symmetric Fault-Tolerant Multiprocessor," IEEE Transactions on Computers, Vol. C-24, No. 5; May, 1975.
22. Pierce, W. H.: Failure-Tolerant Computer Design, Academic Press, New York 1965, pp. 78-114.

TABLE I

FUNCTIONAL AREA	CRITICAL CATEGORY				
	TAKE-OFF	CLIMB	CRUISE	DESCENT & HOLD	APPROACH & LANDING
(1) <u>Navigation and Guidance</u>					
A. <u>CURRENT AIRCRAFT</u>					
a. Area Navigation (R-Nav)	1	2	2	2	1
b. Inertial Navigation System	1	2	2	2	1
c. Autoland (Autothrottle)	0	0	0	2	3/4
d. Radar Altimeter	0	2	2	2	3/4
e. Autopilot	0	2	2	2	3/4
B. <u>1980's AIRCRAFT</u>					
a. Digital Autopilot with Category IIIB	0	2	2	2	3/4
b. Hybrid Navigation	1	2	2	2	1
c. 4-D Flight Path Control	1	2	2	2	3/4
d. Heads-Up Display	0	0	0	3	3/4
(2) <u>Stability and Control</u>					
A. <u>CURRENT AIRCRAFT</u>					
a. Autopilot: Yaw Damper	0	2	2	2	1
B. <u>1980's AIRCRAFT</u>					
a. Cockpit Displays - Electronic Attitude Direction Indicator	3	3	3	3	3
b. Cockpit Displays - Electronic Horizontal Situation Indicator	3	3	3	3	3
c. Flight Envelope Limiter	3	3	3	3	3
d. Ride Improvement System	1	1	1	1	1
e. Limited Scale SAS (Relaxed Static Stability)	3	3	3	3	3
f. Limited Scale Gust and Maneuver Load Alleviation	3	3	3	3	3
C. <u>1990's AIRCRAFT</u>					
a. Full Scale SAS (Negative Static Stability)	4	4	4	4	4
b. Full Scale Gust and Maneuver Load Alleviation	4	4	4	4	4
c. Full Scale Flutter Mode Control	4	4	4	4	4
(3) <u>Air Traffic Control</u>					
A. <u>CURRENT AIRCRAFT</u>					
a. Weather Radar	1	1	2	2	1
b. Ground Proximity Warning System	0	0	0	2	3
B. <u>1980's AIRCRAFT</u>					
a. Digital Weather Radar Processing	1	2	2	2	1
C. <u>1990's AIRCRAFT</u>					
a. Digital Data Link	1	3	3	3	3
b. Airborne Traffic Situation Display	1	3	3	3	3
c. Collision Avoidance System	1	3	3	3	3
(4) <u>Aircraft Systems Management</u>					
A. <u>CURRENT AIRCRAFT</u>					
a. Fuel Control (Tankage)	0	3	3	3	1
b. Fuel Control (Engines)	3	3	3	3	3
c. Engine Inlet Control	1	3	3	3	1
d. Center of Gravity Indicator (Pre-flight)	0	0	0	0	0
e. Weight Monitor (Pre-Flight)	0	0	0	0	0
f. Cabin Environment	1	3	3	3	1
g. Air Data System	3	3	3	3	3
h. Master Caution System	3	3	3	3	3
i. Fire Warning System	3	3	3	3	3
j. Power Generation and Distribution	3	3	3	3	3
k. Automatic Braking System	3	0	0	2	3
l. Aircraft Integrated Data System (Maintenance)	1	1	1	1	1
B. <u>1990's AIRCRAFT</u>					
a. Integrated Data Management System	1	3	3	3	1

## DEFINITION OF CRITICALITY OF FUNCTION

0	1	2	3	4
OFF	NON-ESSENTIAL	REDUCED OPERATING PERFORMANCE; INCREASED PILOT WORKLOAD	POTENTIAL SAFETY HAZARD IF UNATTENDED	INSTANTANEOUS SAFETY HAZARD; REQUIRES IMMEDIATE ATTENTION

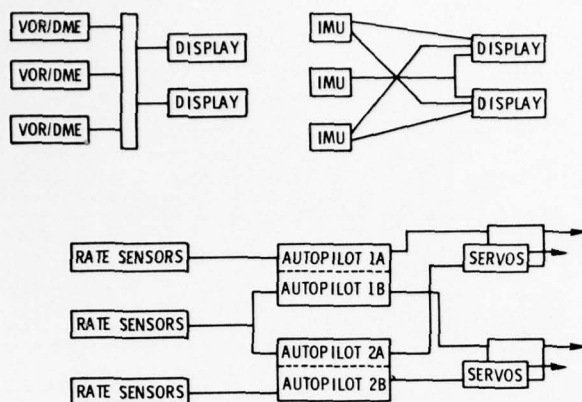


Figure 1. Examples of Contemporary Redundant Subsystems.

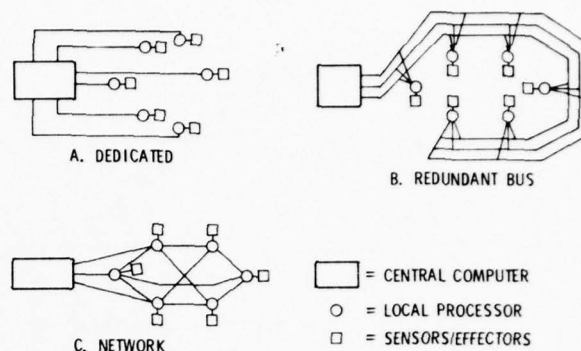


Figure 2. Representative System Structures.

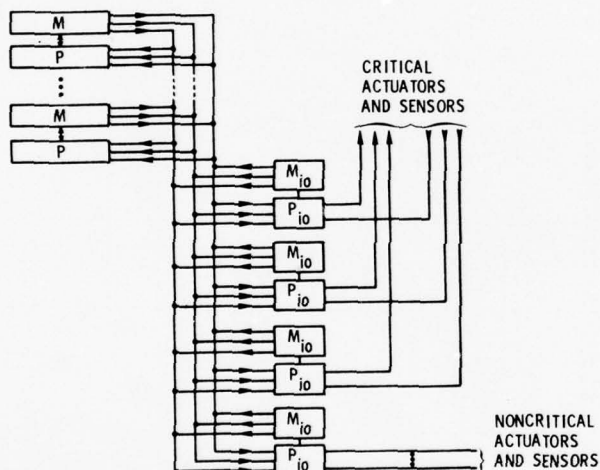


Figure 3. SIFT Configuration.

	PROCESSORS						
	1	2	3	4	5	6	...n
A	X	X		X			
B			X		X	X	
C		X					
D	X		X	X			
E		X		X	X		
F			X		X	X	
G	X	X	X				
H	X			X		X	
I	X		X		X		
J	X	X	X	X	X	X	
...							
N							

Figure 4. Example of Task/Processor Allocation.

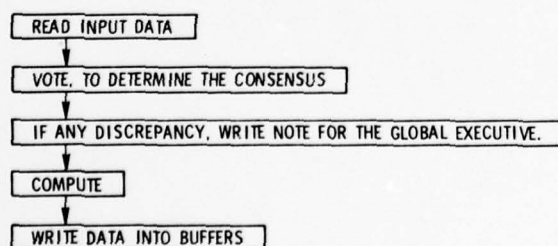


Figure 5. Computational Scheme in SIFT.

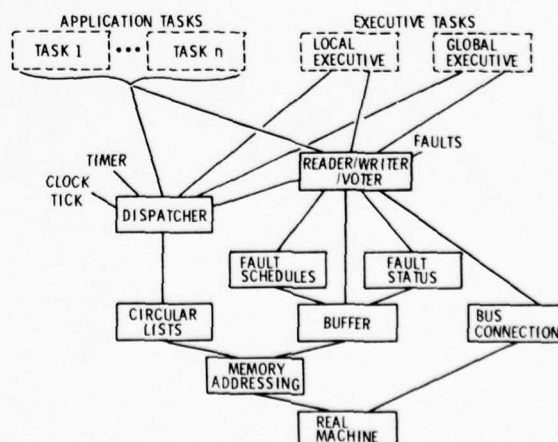


Figure 6. Hierarchical Structure of SIFT.

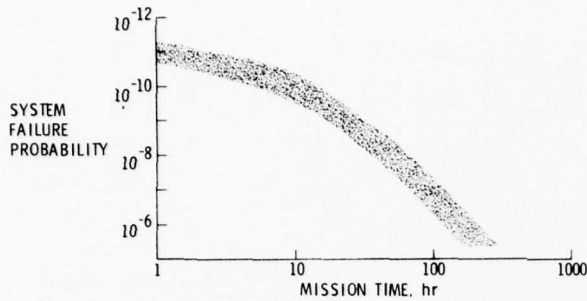


Figure 7. SIFT Failure Probability.

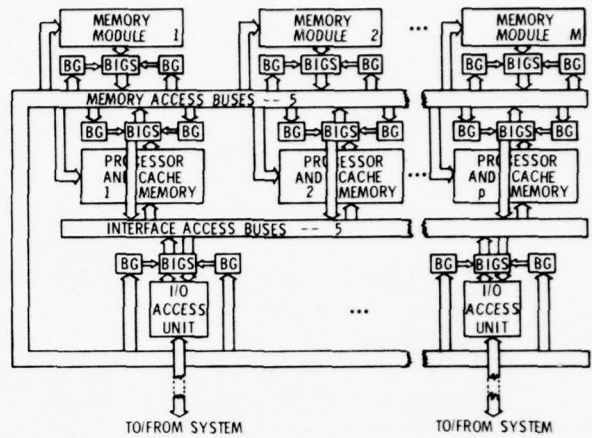


Figure 8. Physical Diagram of Multiprocessor.

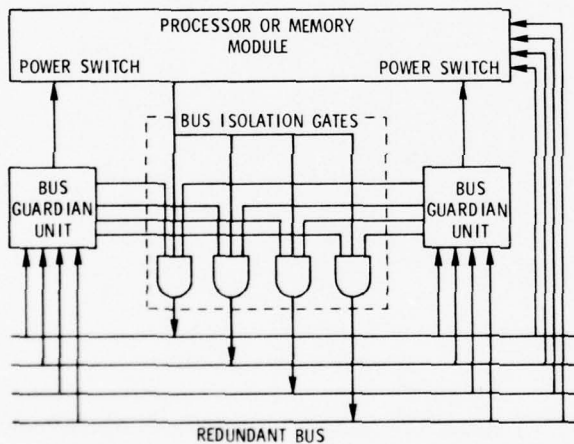


Figure 9. Bus Guardian Connections.

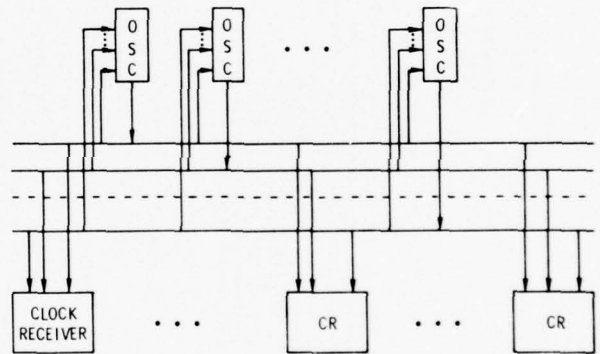


Figure 10. Fault-Tolerant Clocking System.

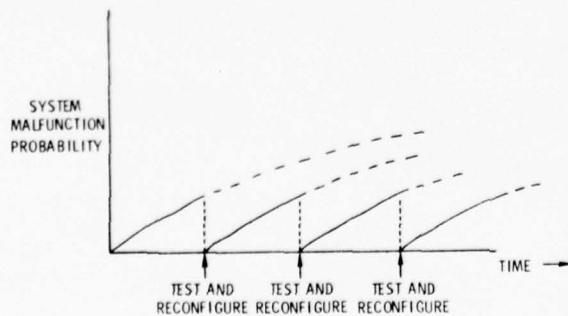


Figure 11. Conceptual Restoration of Reliability by Tolerance Renewal.

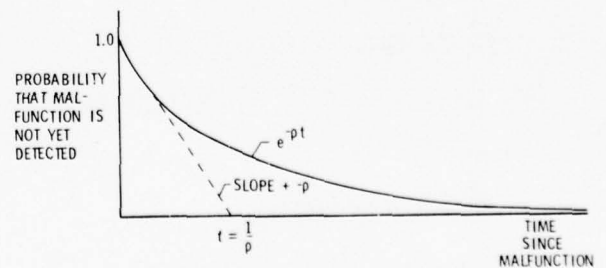


Figure 12. Probability Distribution Assumed for Malfunction Detection Process.



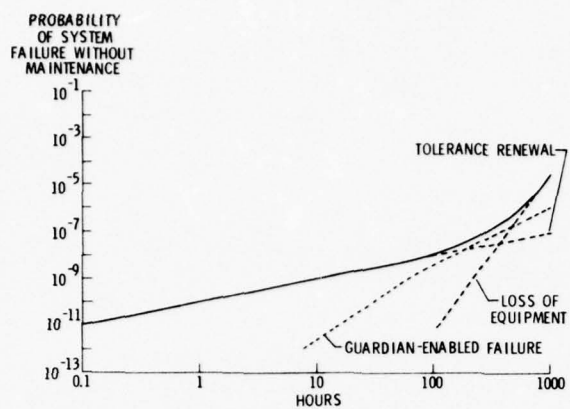


Figure 13. Composite Failure Probability of the Multiprocessor.

## OBJECTIVES FOR THE DESIGN OF IMPROVED ACTUATION SYSTEMS

B. H. EARLEY

CONTROL SYSTEMS DEVELOPMENT BRANCH  
AIR FORCE FLIGHT DYNAMICS LABORATORY, WPAFB

### ABSTRACT

Actuation system reliability is particularly critical to the success of modern Flight Control System (FCS) techniques which utilize high authority electrical controls. The reliability combined with typical performance requirements have a substantial effect on the actuator resulting in a high level of sophistication, complexity, attendant high cost and reduced maintainability. As background, a description of actuation systems evolution and several redundant system approaches are included. Significant redundant system criteria are discussed followed by a series of design objectives pertinent to relief of typical actuator problems. A basic design approach is also recommended to assure timely integration of the FCS in the vehicle design process and proper interfacing of the primary technical disciplines. The implementation and benefits of actuation system development programs are discussed and several new actuation concepts are mentioned.

### INTRODUCTION

With the application of high authority, closed-loop electrical control techniques to primary Flight Control Systems (FCS), the impact of reliability requirements upon actuation systems is greater than ever. Control Augmentation Systems (CAS) and Fly-By-Wire (FBW) in particular utilize these techniques which demand redundant control loops and hence redundant actuation channels. Typical mechanical primary control systems are relatively straightforward and adequate reliability can be obtained through conservative design practices, dual load paths, etc. On the other hand, the sophistication of electrical control concepts and redundant actuators presents a real challenge to be overcome in the process of assuring effective reliability. As with the mechanical approach, reliability of the electrical primary control system is essential to both flight safety and mission effectiveness.

The impact of redundancy on secondary actuation has been particularly significant. For each electrical input signal path, a single stage electrohydraulic transducer provides a hydraulic signal to one or more hydraulic amplification stages producing a final mechanical output which drives the main control valve of the surface power actuator (in conjunction with a mechanical input, if it exists, as in the CAS approach). Proper operation of all secondary actuator stages is dependent upon precision electromechanical or hydro-mechanical components which are expensive. Multiplication of each stage by three or four, the number of channels usually required at least where there is no mechanical input as a backup, results in an impressive amount of complexity and cost. Complexity and precision components similarly increase supply logistics costs and can decrease maintainability.

Stringent FCS performance requirements are dictated by the need to develop aircraft which are superior to those available to a potential aggressor. Modern FCS require extensive interfacing and each application has been sufficiently unique to result in little or no commonality. These factors, in addition to reliability requirements, have collectively been the prime drivers leading to current actuator systems sophistication, complexity and cost.

Due to recent FCS technical advances leading to the application of CAS and FBW systems, the potential for FCS performance growth is unusually high. Simultaneously, the full realization of all benefits will require effective redundancy management, cost controls and improved maintainability - with little or no compromise to the performance potential. An awareness and appreciation of these factors, in addition to a comprehensive understanding of the basic FCS technology, will be prerequisite to the design of a truly successful actuation system.

This chapter focuses exclusively on the actuation system as does no other section of this document. It is significant to note that with the exception of flight control computers, the cost of actuator assemblies can exceed the cumulative cost of the balance of all other FCS components and assemblies. Also, they are most vulnerable to the impact of other criteria, such as reliability for example.

## ACTUATION EVOLUTION

Typical of primary actuation systems are those that control the pitch, yaw and roll axes via elevator, rudder and aileron surfaces respectively. Depending on the aircraft, other surfaces or combinations thereof may be utilized for primary control functions. Spoilers are employed to decrease wing lift and to augment roll control, a stabilator, or all moving horizontal tail surface, to provide pitch control, and elevons, a combination of ailerons and split elevator surfaces to provide pitch and some degree of roll control as well. Leading and trailing edge flaps, trim devices, speed brakes, etc. are not generally considered to be primary flight controls.

Primary control surface power actuators are usually hydromechanical servomechanisms controlled directly by the pilot or through electrically oriented systems. The power source is a remote control hydraulic system (or systems). They are essential FCS elements which in their most elementary form would basically consist of a control valve, cylinder and piston combination. On the other hand, when secondary actuation functions are included, a ship's set may represent 50% of the total FCS cost (exclusive of computational elements). The power actuators are capable of a bi-directional force output, usually linear, or occasionally rotary, which is relatively large in contrast to the actuator size and weight. Output displacement is proportional to input signal magnitude, and output force developed must be compatible with the opposing (aerodynamic) load, which is usually in direct proportion to the displacement. A variable rate is also provided as functions of control valve opening and actuator load.

With a mechanical input/feedback system, displacement of the pilot's control proportionally displaces the actuator control valve causing actuator or piston rod displacement. The actuator force required to drive the surface is generated by metering a relatively high pressure volume of fluid through the control valve to one side of the actuator piston and relieving the relatively low pressure volume on the other side. Motion continues until the rod is repositioned at the point where the mechanical feedback re-nulls the control valve at neutral. The linear output of the unit is translated into rotary surface motion via a crank arm. Most often the control surface will have a neutral or faired position, being capable of an approximately equal rotation in either direction, e.g.  $\pm 25$  degrees.

### Manual Operation:

Prior to World War II primary control surfaces were operated manually except for very special cases. Where high aerodynamic forces were present, motion of the pilot's control, through a mechanical path of cabling, push-rods, bellcranks, etc., drove a surface tab (similar to a trim tab) in lieu of driving the surface directly. (See Figure 1) Surface tab motion, the reverse of the surface motion desired, created a driving moment, forcing the surface itself in the proper direction. Hence, aerodynamic forces acting on the tab, which was integral to the surface itself, created the muscle positioning the control surface to achieve the required aircraft attitude or response.

A variety of other aerodynamic balancing devices were also used to limit the forces which must be overcome by the pilot. Except for these devices, built in mechanical advantage or other assisting mechanisms all forces applied to the surface were derived solely from the pilot. These systems were force limited, slow and completely dependent on pilot input; however, they were inherently reliable, due in large part to their straightforwardness.

### Hydraulic Boost Operation:

The next step in actuator development was to incorporate a hydraulically powered boost cylinder between the pilot and the control surface (normally immediately adjacent to the surface) to supplement the pilot initiated forces to the extent required to drive the surface. (See Figure 2) Hydraulic power was selected primarily due to its force vs weight effectiveness. The pilot's input was connected to the surface crank arm but also moved a valve controlling the boost cylinder piston. Cylinder output force was applied to the same surface crank arm attached in turn to the control surface, hence augmenting the pilot's input force. This mechanization provided the potential for higher surface rates, was comparatively simple and reliable; however, it was still fully dependent on pilot input.

### Fully Powered Actuator:

A more sophisticated version of the boost cylinder was the fully powered mechanical input/feedback hydraulic servoactuator. (See Figure 3) As above, pilot input controlled valve position determining the direction of flow and rate to the hydraulic cylinder, except that there was no parallel mechanical path to the surface. Accordingly the system was irreversible, not permitting any load feedback from the surface to the cockpit, thereby avoiding load non-linearities (reversals, etc.) such as occur in the transonic flight region. This approach also unfortunately eliminated inherent load sensing feedback necessitating the implementation of artificial feel systems. Although not shown in the figure, a mechanical feedback linkage configuration sensed actuator output displacement so that when surface position corresponded to the input command, the initial valve signal was offset, returning the valve spool to a null position. Pilot control of aircraft response was indirect via sensing of the surface position, i.e. if additional response was desired, the control displacement would be increased requiring additional actuator stroke for the feedback to null out the control valve. This actuator type provided full driving force to

the surface, inherent irreversibility and more positive and accurate control.

#### Multi-Input Actuator:

Next, during WW II, the ability for the actuator to respond to an electrical input autopilot signal was added. An electrohydraulic servo consisting of an Electro Hydraulic Valve (EHV) controlling a hydraulically powered secondary actuator (or servo) was located in parallel with the mechanical control path. (See Figure 4) The low level autopilot input to the EHV was converted to a hydraulic signal driving the secondary actuator piston producing a proportional mechanical signal of sufficient strength to mix with the pilot's mechanical input. The arrangement was such that the autopilot electrical input not only moved the power actuator control valve, but the entire linkage system as well. This provided the pilot a means of monitoring the automatic inputs through motion of the controls. Since autopilot corrections required during cruise were small, the secondary servo authority was limited giving the pilot capability to override the inputs and avoiding the necessity for redundancy.

#### Stability Augmentation Systems (SAS):

For two basic reasons SAS came into being. First, due to the differences in aerodynamic loading and response, it became impossible to obtain good flying qualities in all flight regimes, particularly at both ends of the speed range. Secondly, inherent stability varied considerably as aircraft approached and exceeded the speed of sound. Vehicle designs exhibited marginal stability under certain conditions; therefore, another type of automatic system was required to compensate. SAS implementation is typical of the configuration shown in Figure 5 utilizing a secondary hydraulic series servo in the mechanical path. Rate gyroscopes sensed aircraft oscillations which through the actuation system could damp out the undesired motions by the control surfaces. Electrical inputs from the gyros drove the secondary servos in the same manner as the autopilot; however, the series signal interface was such that the SAS input added to or subtracted from the pilot input and was not reflected back to the cockpit controls. Similarly, SAS authority was limited and redundancy was usually not employed. Inherent disadvantages of this mechanization were that it could reduce pilot authority and could not differentiate between pilot and disturbance inputs.

#### Control Augmentation:

To avoid the problem of the automatic system detracting from control effectiveness, an electrical feedforward path, sensing pilot control inputs could be added to the Figure 5 configuration and summed with the airframe dynamic inputs such as the rate gyro and accelerometers. With this addition, as shown in Figure 6, the control loop could be closed via an electronic system so that control inputs would be compared to actual vehicle response rather than to a surface position. Thus a third basic type of mechanical/electrical input system, the Control Augmentation System (CAS), was derived. With this mode of control the primary path was electrical in nature with the mechanical path utilized as a backup. Accordingly, the level of authority possible through the secondary servo was increased up to 100%, necessitating redundant techniques. At this point the technology and equipment reliability available were such that primary control via multiple channels was practical, however, the potential to significantly complicate the system was introduced.

The Figure 6 diagram does not show an independent (or remote) secondary actuator, but is typical of packaging to incorporate both secondary and power actuator functions. Either parallel or series arrangements could be accommodated within a single assembly. On the other hand, the actuator as a single Line Replaceable Unit (LRU), became complicated and expensive, particularly so since redundant channels were required. If the secondary and power actuator functions were separated, the vast majority of the cost and complexity would simply be shifted to the secondary unit since this is where the actuator redundancy must be concentrated. Also, the extent of actuator interfacing required is more than doubled.

Electrical input capability has provided tremendous advances in flight control technology. It automatically controls the aircraft during cruise, compensates for subsonic and supersonic flight conditions, offsets stability limitations and actually improves vehicle control by providing more responsive and accurate control than would otherwise be possible. With the advent of Fly-By-Wire (FBW) the near future offers much more to be exploited.

#### Fly-By-Wire:

The complete elimination of the mechanical control path results in a FBW control system where all commands to the actuators are transmitted electrically utilizing high authority closed loop techniques. Vehicle dynamic feedback again closes the control loop so that airplane response rather than surface position is directly controlled. A typical implementation is shown in Figure 7. Today the state-of-the-art has progressed to a point where confidence due to experience with electrical input techniques and the reliability of available equipment is adequate to justify the full FBW approach. Pseudo FBW systems, where a normally disengaged mechanical path is retained for backup, are still popular and the first pure FBW operational aircraft has yet to enter the active inventory. Retention of the mechanical controls as a backup is a significant disadvantage because of its associated cost, attendant complexity, rigging requirements and the fact that a comprehensive declutching scheme is needed. Where it is necessary for the mechanical and electrical paths



to operate simultaneously, as with the CAS, both must be closely synchronized to minimize mechanical anomalies which would adversely effect the function of the electrical primary path. For example, mechanical hysteresis could introduce limit cycling, or underdamped system oscillations, making precise operation of the mechanical elements mandatory.

Exclusive of the considerable potential for FBW derivative concepts and the increased opportunity for major systems integration, the basic FBW approach offers many advantages. These are an improved level of precise performance, improved reliability through redundant techniques, reduced weight and space requirements, reduced design, installation and maintenance and finally, reduced life cycle costs. With the elimination of the mechanical path, overall system complexity is reduced; however, with the attendant need for redundancy, the actuator, particularly the secondary assembly remains quite complex.

#### Current State-of-the-Art:

Today's flight control systems are sophisticated, high performance systems capable of enhancing vehicle performance as well as providing basic control. FBW technology is now about to open a new era in flight controls. It has the potential not only to significantly affect basic aircraft design and configuration, but also to remove long-standing constraints, for example the need for a complete mechanical control path and complex feel systems. Most important, FBW will provide the catalyst to broaden controls application and the functions which may be accordingly performed, substantially increasing the flexibility and usefulness of the hydraulic servoactuator, but increasing the demands it must satisfy. Typical new technology applications include alleviation of structural loads, attenuation of aerodynamic flutter, flight path smoothing, safe control with reduced static stability, and maneuver load control. Maneuverability and performance is enhanced by systems providing Direct Lift Control (DLC) and Direct Side Force Control (DSFC). All of the associated benefits are facilitated, if not made possible, via FBW technology.

Integration of the FCS with other major aircraft systems is now much more practical. As an example, the engine controls may be coupled with the flight controls to provide improved and automatic intersystem coordination during certain flight modes such as take-off and landing. Or, particularly applicable to fighter aircraft, the flight controls may be integrated with the armament system to properly aim the fuselage to assure accuracy in weapons delivery. Tracking systems can be readily coupled to the FCS to maintain the attacking aircraft in firing position throughout maneuvers initiated by the target aircraft. It is further practical today to interface the primary flight control system to an electronic terrain avoidance system allowing low altitude, high speed penetration of bombers or fighters.

Finally, the effectiveness and efficiency with which the complete aircraft mission is carried out can be maximized via airborne computer systems. These systems are programmed to cause the flight control system and the aircraft in turn to perform the desired functions in accordance with predetermined mathematical approximations of the corresponding inputs required. The mathematical approximations are multimode control laws which allow the flying qualities to be modified to produce nearly ideal dynamic characteristics tailored to specific mission tasks. (Ref 1)

All of the above are new tools which the designer may utilize to satisfy aircraft control requirements more effectively. Many of the types of applications named are new and unique and most have at least been test flown; however, the technology is essentially untried in operational fleet aircraft. Once matured, the FCS improvements realized should be dramatic.

#### Redundancy Management Concepts:

Although the idea of redundancy is not new to flight controls, the demands made by the introduction of FBW are so substantial that the subject of FCS redundancy management for modern aircraft is a new challenge of tremendous complexity. This is so much the case that several aerospace companies have undertaken comprehensive laboratory development programs primarily for educational purposes. These programs, frequently internally funded, are appropriately considered by management as necessary to the company's competitive status by providing a firm technical base for future programs. Companies with actual FBW experience also benefit from this approach by expanding their backgrounds and by obtaining maximum system data prior to flight test.

The typical servoactuator will incorporate several visible levels of redundancy. A popular power actuator configuration is the dual tandem type having two isolated in-line cylinder/piston chambers controlled by a one piece dual tandem valve. Like the valve, the two pistons and the rod which transmits the actuator output force will be a single structural entity. The mechanical simplicity and predictable failure modes, combined with appropriate design conservatism, typically enable this straightforward approach. Hence the output itself is generally a single load path; however, the valve and power actuator are dualized to allow for continued operation at 50% output subsequent to failure of one of the two hydraulic power sources.

Continuing the description of the actuator from output to input, the major impact of redundancy is essentially limited to the hydromechanical secondary actuator. Each independent redundant channel which is required to provide a given overall level of reliability adds to the total complexity and number of precision parts. It is at this point that the designer's task becomes most comprehensive. There will generally be three or four separate and equivalent electrical channel inputs of relatively low signal strength

to the secondary actuator. How these signals are combined and utilized before and after failure(s) is dependent on the redundancy concept selected. Frequently after having sustained two failures the system must still be operative, reverting to a fail safe or neutral condition on the third failure. This is a Double Fail Operate/Fail Safe (DFO/FS) system. Not all dual failure conditions may be accounted for, particularly if the possibility of their combined occurrence is less than remote.

Where not all dual failures would permit continued operation, strictly speaking the system would be classified as Single Fail Operate (SFO). Equally important is the level of performance degradation which can be tolerated after each failure. For example with a DFO/FS system it may be practical to complete the aircraft mission, while with a SFO system, return to base would be imperative.

Although numerous derivatives exist, there are probably less than half a dozen basic redundancy schemes. Several popular types will be described beginning with the force sharing approach where the outputs of each individual secondary (or power) actuator channel are structurally combined, driving the load collectively.

#### Force Sharing:

An example of force sharing system was the philosophy applied to the American Supersonic Transport (SST) where the basic concept was applied to all primary surface power actuators, to the secondary actuators or Electric Command (EC) servos and to Master Servo (MS) units. In the pitch axis, shown in the simplified schematic of Figure 8, a highly reliable electronic stability augmentation system was required for safety of flight. Four redundant electronic channels were necessary to assure that the loss of function resulting from multiple failures would be an extremely remote possibility. The primary mode of control was electrical with a single load path mechanical system providing a backup following a multiple failure. Safety and high electronic systems integrity dictated the utilization of two functionally different electronic systems, the Hardened Stability Augmentation System (HSAS) and the Electric Command and Stability System (ECSS). The HSAS provided pitch axis stabilization to ensure minimum safe handling qualities and the ECSS provided normal actuation system control and aircraft handling qualities.

Pilot commands were transmitted mechanically through a single cable system to a three channel Master Servo (MS) unit which functioned as a force isolating boost servo and was connected mechanically to the control valves of four individual stabilizer actuators. The cable system was able to be non-redundant since this path provided a backup function only. Accordingly the feel mechanism was implemented at the control columns and a spring centering device for the MS units, to ground the mechanical system following cable failure, was located at the opposite end of the cable system. Multiple load paths were provided between the control columns and the feel system and between the MS units and the power actuators.

Pilot commands were transmitted electrically from four control column mounted transducers through both the HSAS and ECSS systems to the Electric Command (EC) servos which were integral with the power actuators. EC servo output was summed with the MS (mechanical) output immediately upstream of the actuator main valves.

Since the cable (mechanical) system input was non-essential, total surface command capability was necessary for the electrical control mode. During normal operation mechanical input was sensed by a negator transducer and fed back to the system electronics where it was summed with the HSAS and ECSS system outputs. Accordingly the mechanical system input was subtracted from the EC servo electrical input so that the servo output was equal to the difference between the desired command and the mechanical input.

To provide the required level of reliability, both the HSAS and ECSS were four channel, powered by independent electrical sources and all hydraulic elements were supplied by separate hydraulic systems. Complete HSAS channel separation was required permitting no interchannel connections or cross channel voting. ECSS system gain scheduling and cross channel voting were incorporated, but the additional complication was considered non-essential to the HSAS and likely to compromise inherent reliability and simplicity. The ECSS was the only system non-essential to safety which interfaced with the HSAS; however, ECSS implementation was such that its failure could not effect HSAS operation.

Each EC servo output, in addition to driving the valve linkage of the actuator with which it was integral, was also interfaced with the other EC servos by an external synchronizing shaft. Anti-jam detents prevented a single EC servo failure from affecting more than one actuator. When such a failure occurred, depressurization of the faulty actuator allowed continued operation on the remaining three channels.

EC servo equalizing bypass valves were utilized in each servo to minimize input offsets and tolerances between the electrical channel signals and to avoid switching transients. Since the EC outputs were force summed, equalization was implemented by Linear Variable Differential Transformer (LVDT) feedback from the valves to balance the steady state individual channel inputs. An electrical integrator with a pre-determined threshold was incorporated to avoid eliminating the steady state force capability of the servos. (Ref 2)

In the yaw and roll axis, rather than integrating the EC servos with the power actuators, the four EC channels were packaged together, with an integral synchronization shaft, as a single Line Replaceable Unit (LRU). One EC servo package, mounted separately and interfaced mechanically, was provided for electrical control via the ECSS of each group of surface power actuators. Identical MS units were interfaced as in the pitch system.

With the exception of the pitch EC servos, all other units were identical, and as in the pitch system had DFO/FS capability. Each EC servo piston was monitored by an LVDT which following a failure provided an output feedback causing the remaining good channels to oppose the failed channel and minimize failure effects. Once failed, a channel would be depressurized, the piston bypassed and the synchronization shaft detent released. Degraded, but adequate operation was practical following failure in any two channels. As each failed EC channel shaft detent was released, a ground bias load was imposed on the remaining channels. Upon failure of a third EC channel, the remaining channels would be shut down and the collective bias load provided an effective ground point for control through the mechanical path. Hence the EC servos failed safe and mechanical reversion control (without HSAS) was practical for all except aft c.g. locations.

The MS units each utilized three hydromechanical channels (with three hydraulic sources) all receiving the same mechanical input through a cable system from the pilot controls. With the external spring centering device, which caused the three MS to return to a neutral position following a cable failure, and since there were no provisions for selective channel shutdown, the redundancy level was SFO/FS. The MS units were strictly hydromechanical; therefore any servo channel differences were force averaged at the output summing point.

All primary control surfaces were driven by groups of three power actuators with the exception of the stabilizer and lower rudder which used four and the upper rudder which used two. Output levels were selected so that four actuator groups could develop 133% and three actuator groups 150% of the surface design hinge moment. Accordingly, following a single channel failure, depressurization and subsequent bypass of the channel would allow essentially 100% of the design value to be developed. Normal operation of the actuator groups, like the MS units, was in a force averaging mode. Operation of the EC servos, due to the equalizer feedback concept employed, was typical of a majority voting mode.

#### Active Standby:

A redundancy concept frequently utilized in applications where completely independent electrical and hydraulic sources are limited is that popularly known as active standby. The concept utilizes a two or three channel secondary actuator with the outputs mechanically bussed together; however, as implied only one channel controls the load since the standby channel piston(s) are bypassed hydraulically and hence in a standby mode. Although bypassed, the standby channel piston(s) track the active piston, the EHV(s) receive an equivalent input signal and accordingly track the active channel EHV. Generally, each channel would include an electronic model of the EHV control loop for comparison with actual EHV output obtained via an LVDT monitoring the valve second stage output. LVDT feedback during normal operation will limit differences between the active and standby channels to minimize transients should reversion become necessary. Upon detecting an output difference above a pre-determined limit in the active channel, an electronic comparator output would then initiate a correction sequence disabling the channel and activating a standby channel. The failure detection/correction logic may also be implemented hydro-mechanically although this may be a more expensive alternative due to the precision components required.

The actual level of redundancy obtained will be dependent on the number of channels and the particular mechanization; however, although two channels may be limited to SFO, three channels (one active/two standby) will not necessarily provide full DFO capability. Frequently less remote dual failure protection possibilities are provided for without incurring the complication and expense of accommodating all.

The secondary actuator may be integral to or remote from the power actuator which most often will be a dual tandem configuration. Hydraulic supply failure(s) must also be accommodated by the secondary actuator assembly; however, if only two independent sources are available (frequently typical in fighter aircraft), a single system may be shared between two of three channels or a method may be devised to provide partial separation within one system. These expedients may be necessary in many multi-channel applications.

#### Active-Active:

A modification of the active standby concept involves simultaneous operation of a pair of active channels using the output of each to directly control one half of a tandem power actuator. A third channel may also be conveniently utilized as a hydromechanical model which will more effectively model typical hydromechanical non-linearities as compared to an electronic model. The output of the two active channels may be balanced by a spool equalizer valve to offset any input or EHV induced output differences. Differential output of both EHV's is sensed by the equalizer and subsequent spool position fed back mechanically to the EHV torque motors. Excess spool displacement (above a pre-determined limit) would signal a failure in one of the active channels. Comparison between the active channel outputs and the third channel model enables the hydraulic failure logic to select



and disengage the faulty channel. The actuator then continues to operate at one-half of the normal output level. A failure in the model channel does not affect active channel operation, but upon detection provides an electrical output for indication to the pilot.

This type of mechanization is typical of the F-111 CAS actuator. The package and its failure logic are integral and it is remote from the power actuators. Mechanical pilot input is interfaced at a point between the CAS and the power actuators. The aircraft utility hydraulic system supplies one active and the model channel and the primary hydraulic system supplies the remaining active channel. The level of redundancy is SFO, although continued operation is possible following a number of specific dual failure occurrences. (Ref 3)

#### Active-On-Line:

A final variation of active standby to be discussed is active-on-line which may have considerable near future application. The primary difference exhibited by this approach is that the on-line (standby) channel pistons are not bypassed and track the active channel of their own accord rather than being forced by it. Active channel control is assured by a high gain negative force feedback which limits the output force of the on-line channels. On the other hand, the on-line channels normally operate in a force sharing mode and will immediately oppose a failed channel. The failure logic would incorporate an electronic model in each channel with inner and outer loop comparitors. Following detection of a signal mismatch above pre-determined limits in the active channel, a sequence would be initiated, after an appropriate time delay, to disable the active channel and cut-off the feedback circuit in the first on-line channel making it active. Initial opposition to the failure occurs without intervention of the detection/correction logic and its associated time delay. Pressure feedback is utilized to limit errors between the active and on-line channels, therefore minimizing switching transients. With three channels, DFO capability is practical. This concept may prove to be most tolerant of dual failures, having the least adverse effect on post failure performance.

#### Redundancy Criteria:

Proper redundancy management is integral to the achievement of adequate flight safety and mission effectiveness for high authority CAS and FBW systems. It is also essential to effective cost control, successful design, implementation and operation, and is frequently the most difficult technical challenge facing the actuator designer. The approach to be taken and the extent of redundancy necessary to meet specified reliability levels actually determines the mechanization and the extent of complexity of the secondary actuation system elements. The significant factors are:

- a. Criticality of the function to be performed to flight safety and mission success. If after one or more reasonably probable failures, including hydraulic supply failure, continued operation of a control surface is essential, then appropriate actuator redundancy must be incorporated to assure given performance levels after each occurrence. Generally electrical control loop reliability must be equivalent to that of a primary mechanical path.
- b. Determination of the basic control modes. Is the primary control mode to be electrical or mechanical and is a dissimilar back-up mode to be incorporated? If electrical, the control authority will usually be 100% and the greater the authority, the more comprehensive the redundancy must be. A mechanical backup may reduce the extent of secondary actuator redundancy; however, it also introduces electrical/mechanical interfacing problems.
- c. Type of aircraft and its configuration. Typical variables are: the type of aircraft and mission; power source availability (hydraulic and electrical), space/weight limitations, interactive effects between control axes, the degree of integration required, etc.
- d. The level of reliability of individual components and equipment items in the system. The effect the reliability of individual items in the system is relatively straightforward. Reliability in general, particularly of electronic equipment is constantly improving, which facilitates achievement of the overall system goal.
- e. The degree of system integrity enhancement necessary to assure confidence in the FBW approach itself. FBW is a major technology breakthrough which is understandably difficult to accept as an adequately reliable complete substitute for the familiar mechanical system. Accordingly there may be a tendency to incorporate too much redundancy which in actuality will add complexity and cost and increase failure probabilities. It is well to note here that each active electrical signal path requires an independent secondary actuator channel. The point at which the redundancy level is effective and not excessive is not readily apparent; however, the designer should exercise the utmost caution to avoid possible compromise of reliability while attempting to provide additional insurance.

Redundancy level and approach have a direct impact on actuation, particularly on secondary actuation, and few FCS reliability related considerations may be made independently. Once the redundancy level and approach have been determined, the secondary actuator is basically defined. In addition, failure detection/correction, feedback implementation, transient protection, freedom from inadvertent channel shutdowns (when no actual failure has occurred), control of failure degradation, etc. are functions which



must be accommodated within the actuator design.

Accordingly, it may be observed that the design and implementation of an effective actuation system redundancy management concept is a very comprehensive technical task. It is essential to incorporate redundancy with the FBW system concepts which offer so much potential to near future aerial warfare capability. The capability exists to achieve required reliability within reasonable risk and cost limitations. The degree of success realized will be dependent on adequate background, design flexibility and appropriate priority.

#### ACTUATION SYSTEM DESIGN OBJECTIVES

The following are suggested not only to obtain actuation system reliability, but effective system performance, reasonable life cycle cost and improved maintainability as well. This material is based on Air Force Flight Dynamics Laboratory (AFFDL) actuation system related experience, particularly FBW associated programs, obtained through development efforts dating back approximately twelve years. These recommendations are broad in nature and do not reflect the many detailed considerations which must be made in the process of designing a successful state-of-the-art system.

##### Reliability:

To a CAS and particularly to a FBW system, adequate reliability is a most critical factor since it is inherent to flight safety and mission effectiveness. Excluding the back-up system approach, application of some redundancy concept to the secondary actuator is the sole method by which the needed reliability can be practically obtained. Selection of a basic concept and determination of necessary redundancy levels are the major decisions to be made relative to definition of the actuator mechanizations. A sound and broad background of information, preferably based on development program or actual experience will be pre-requisite to making the proper decisions. Less than optimum resolution of these questions, in addition to compromise of safety and effectiveness, may also adversely affect life cycle costs, complexity, size and weight. Thoroughly evaluate the data available from all sources in conjunction with the previously discussed redundancy factors, then design the actuation system mechanization to meet the requirements and enhance overall reliability, rather than detracting from it.

##### Life Cycle Cost:

Historically the actuation elements of a FCS have been relatively expensive; however, present state-of-the-art sophistication can tend to aggravate the problem. Obviously the FCS is a major aircraft system and can have an associated major impact on the total weapon systems cost. Included in the life cycle cost are not only acquisition costs (development, design, implementation and test) but operational logistics and maintenance costs throughout the lifetime of the system. The impact of high costs is to limit weapon systems procurement in some cases, if not result in cancellation of an entire program as has happened. With the demands placed on the actuation system by performance, reliability and other requirements, actuator associated expense is not surprising particularly in consideration of the precision electromechanical and hydromechanical components necessary. A number of the below listed problems also constitute contributing factors. Although a truly low cost actuator appears to be an unobtainable goal, substantial cost effectiveness improvement is entirely practical. Accordingly, based on thorough evaluation, complete and reasonable actuation system cost objectives should be established. The system should subsequently be designed for compliance with these objectives.

##### Complexity:

The causes of actuation system complexity are nearly synonymous with those of high cost; where excessive complexity exists, so will excessive cost. Complexity seriously compromises the achievement of effective reliability, for example, as the number of components increase the number of potential failures which could occur will similarly increase. Excessive complexity can also adversely affect performance, mission effectiveness and flight safety. Due to typically high performance level requirements, redundancy and interfacing requirements, the military aircraft in particular will always exhibit a relatively high level of complexity; however, all resources which may be applied to limit complexity, while satisfying performance and other criteria, will be well spent. It is also pertinent to note that the most straightforward design may also produce the highest level of integrity.

##### Performance:

Actuation system performance is of primary importance to mission effectiveness of the vehicle; therefore, it must be consistent with overall system requirements. Generally new applications require improved actuator response, rate capabilities, etc. On the other hand, the techniques required are usually well within the state-of-the-art. Frequently better performance simply means more complication and higher costs so that the limiting factor is not technically based, but is cost associated. Understandably, major emphasis is always placed on performance while complexity and associated cost seem to be regarded by many as unavoidable disadvantages. Today because of the high level of sophistication typical of actuation systems and the attendant costs more effort is required to apply reasonable controls. Little or no performance compromise should be incurred in the process and the benefits of reduced costs and complexity will be returned throughout

the life of the weapons system. The need for aircraft which are practical to buy, operate and maintain is nearly as predominant as the need for superior performance.

#### Commonality:

Although the basic redundancy concept to be utilized on a given aircraft assures some degree of inter-axis commonality, considerably more effort can be effectively utilized to design actuators for more universal application. In most aircraft, basic similarity is only typical in the roll axis primary controls where differences are usually limited to those needed for right and left hand applications. Now with the advent of the F-16, the same basic actuator is utilized to drive each horizontal stabilator, both flaperons and the rudder surface. Differences are incorporated to accommodate stroke, force output and rate requirements; however, one actuator and one servovalve specification completely define the three configurations required for all surface applications. Modularization concepts are particularly applicable to actuators. As above, a basic control manifold, or valve module can be designed for impelmentation with various ratings to match different power actuator requirements. The idea of modularization is not new although application to actuation systems is relatively limited. Any practical expedient to promote more universal actuator or other equipment applications should be promoted.

#### Integration:

Frequently the actuation system for a given surface application will be implemented in two physically separated assemblies, the power actuator and the associated secondary actuator. Significant disadvantages with this approach are that mechanical anomaly will be aggravated in proportion to the length of the signal path between the LRVs, the extent of actuator interfacing, i.e. structural, electrical and hydraulic, electrical, may be doubled, and costs will undoubtedly be greater. Hence, where it is at all practical, integration of these major actuation system elements is strongly recommended. F-16 designers also adopted this approach which compared to the non-integrated YF-16 actuators are expected to result in substantial per aircraft savings.

#### Back-Up Systems:

The implementation of a back-up system, which implies application of a dissimilar method by which a given task may be accomplished, may be quite effective for selected functions. Further, back-up mechanical controls may be an absolute necessity where for environmental or other reasons, a primary electrical control system could not be self-sufficient. On the other hand, implementation of a back-up mode will impose disadvantages, for example overall costs will probably be greater, performance degraded, complexity, weight and space requirements will be increased. Accordingly, if at all practical, it is generally better practice to provide the needed reliability within the basic primary system.

#### Sensitivity:

This redundant system characteristic is related to the susceptibility of the control mechanization to certain normal operating conditions which may appear to the failure logic to be an actual failure. An effective actuation system must be designed to avoid these nuisance failures which will cause the actuator to switch to an other than normal mode of operation. Selection of failure detection levels must be made to accommodate system pressure variations and other transient conditions without triggering the failure detect/correct sequence. Inherent tolerances must be accurately estimated and the resulting normal operation band cannot be too restrictive. Frequently a time delay is successfully utilized to reduce the possibility of nuisance tripping. On the other hand, once the alternate or standby mode of operation has been selected, latching provisions should be incorporated to prevent migration back to the non-failed mode.

#### New vs Mature Equipment:

In the design of a new actuation system, some equipment of new design will be required to fulfill specific requirements unique to the vehicle. As compared to the use of existing, mature equipment, design, implementation and test costs will be higher, technical risk is increased and reliability can only be estimated. In view of the inherent disadvantages of new equipment, a concerted effort should be made to achieve an effective mix of new and mature equipment in the system design.

#### Maintainability:

This problem is longstanding; however, the implication of many different and highly sophisticated actuation systems is apparent: maintainability will not improve of its own accord. It too needs appropriate emphasis in the design process if the designer is to produce a relatively straightforward, foolproof and reliable actuator configuration which complies with all requirements. Brief consideration of statistical maintenance data on operational systems documents the need for improvement in this area.

#### Flexibility:

Frequently, some manufacturers tend to apply a standard in-house approach to actuation systems design. Where inadequate knowledge and experience with other alternative concepts exists this may be the sole practical expedient, but one which may easily lead to

reduced system effectiveness, increased complexity or other compromise. To successfully manage and apply modern actuation concepts to a new vehicle, no substitute will be found for a broad and in-depth understanding of available technology. No single approach can be universally applied; however, the experience and expertise which may be derived from a continuing and comprehensive development program will provide the background and objectivity necessary to define the optimum approach.

#### FCS/Actuator Specifications:

There are two generally applicable military specifications which will provide substantial guidance to the actuation systems designer. The first is USAF specification MIL-F-9490, "Flight Control Systems - Design, Installation and Test of Piloted Aircraft, General Specification For." This specification was recently revised extensively in a program sponsored by AFFDL. Revision D was released in June of 1975 and a comprehensive users guide (AFFDL-TR-74-116) has also been made available. The basic document is compatible with modern FCS technology and as mentioned above adopts a systems design approach. General FCS criteria, not covered in detail specifications will be controlled by this document on new AF piloted aircraft procurement programs. The specification and its associated user's guide are quite comprehensive and will prove to be unusually effective, informative and up-to-date.

In addition AFFDL has prepared an actuator specification entitled: "Actuators, Aircraft Flight Controls, Power Operated, Hydraulic, General Specification For." The document is presently in the process of industry review and is expected to be released during 1977. (Military specification number not yet assigned.) Interim guidance is available in a Society of Automotive Engineers (SAE) Publication, Aerospace Recommended Practice (ARP) number 1281 entitled, "Servoactuators: Aircraft Flight Controls, Power Operated, Hydraulic." etc.

The importance of technical specifications cannot be overstated. They assist in definition of the lines of responsibility, provide a basis for arriving at an initial cost breakdown and they describe the physical and other characteristics of package to be delivered. Package performance, reliability, maintainability and required tests and test methods are usually fully defined. The two basic types of specifications, general and detail are each complementary to the other so that a component or assembly, i.e. EHV and actuator respectively, will be required to comply with both and ultimately with the system specification(s), i.e. MIL-F-9490. Detail specifications must accordingly be carefully prepared to comply with specifications for the next higher order of assembly, etc. The designer will acquire a detailed familiarity with these documents and will continue to work closely with them throughout the design, implementation and test processes.

#### Design Approach:

Since the actuation system has a predominant effect on vehicle performance and on the basic vehicle configuration, the actuation design must be given adequate consideration early in the design process. Traditionally preliminary FCS design incorporates the aerodynamic, propulsion and structural disciplines in determination of the initial aircraft configuration. New actuator applications such as direct lift and sideforce control enhance basic controllability, others allow reduction of inherent vehicle stability, provide flutter mode control etc. through automatic actuation systems which can substantially affect configuration and the primary disciplines. Accordingly, the FCS must be considered at an earlier point in the design program so that the interactive effects may be fully defined and best advantage taken of available FCS technology.

Similarly, the increasing trend to integrate the major systems involves another change to the traditional design approach. The integration itself demands extensive interdisciplinary liaison; however, an organizational structure broken down into separate design groups tends to limit the necessary contact since there is little inherent encouragement to cross the existing organization boundaries. Major success in implementing new aircraft having integrated systems and otherwise demonstrating effective application of available technology will be best accomplished via organizational structuring which facilitates and encourages technical interchange. Designers must have a broader background to be able to understand interactive effects on the total vehicle. Revision D of MIL-F-9490, "Flight Control Systems - Design, Installation and Test," etc, emphasizes a total systems approach, reflecting the interdependence of components, equipment and systems in recognition of these changing requirements. (Ref 1)

#### FBW Development Programs:

To effectively manage and apply modern actuation concepts to a new vehicle, no substitute for a broad and in-depth understanding of available FBW and redundancy management techniques will be found. Although a considerable amount of descriptive material exists today, in the absence of sufficient past experience it is likely that a FBW development program would be pre-requisite to optimum design of a new actuation system. Such programs should be comprehensive, continuous and not limited to an investigation of a single approach or concept. Such narrow objectives may have immediate benefits related to specific applications; however, the airframe manufacturer - and the customer - should have a detailed familiarity of all potential approaches. This capability will promote the objectivity necessary to proper selection, as opposed to a continuation of past practices and techniques which could be improved upon.

An adequate development program requires the commitment of substantial funding and manpower resources; however, there are avenues by which these expenditures can be



maintained at practical levels. First, if a program is not initiated on a crash basis, it may be conducted over a period of time with reasonable and relatively level manning. Secondly, joint programs can be an efficient method by which experience and similar objectives can be realized by the parties involved at a much lower cost to both. For example, to support a laboratory actuation system test program, an equipment manufacturer may find it to his advantage to provide the actuator to gain experience with a new concept and establish a better competitive position which could lead to subsequent production. The prime manufacturer conducts the laboratory program, presumably making pertinent test data available to the actuator manufacturer. Thirdly, the military services commonly support development programs as is in their best interest, since they are the ultimate customer. This is undoubtedly a most significant factor with respect to the level of FCS and actuation technology available today.

The approach that actuation system development programs should take must be all encompassing, but obviously is dependent on experience to date. Keeping abreast of available technology and current developments is the prime objective; however, actual experience in the laboratory with redundant FBW actuation system equipment is even more essential to maintaining an adequate background. Studies, analyses and claimed performance may be verified and problems will be encountered and effectively resolved, avoiding the associated expense of later disclosure during flight test, where worse consequences may also be incurred. When the opportunity arises to propose, design and implement an actuation system for a new application, it may then be accomplished efficiently on the basis of a firm knowledge of the factors involved.

#### New Concepts:

Several new approaches having the potential to resolve some current actuator problem areas are now being developed. A most important concept is a direct drive valve mechanization where the actuator main control valve is driven electromechanically. This allows elimination of the intermediate hydraulic amplification stages of secondary actuators, drastically reducing the complexity and cost inherent to the conventional hydromechanical secondary actuation approach.

#### Direct Drive:

Methods evolved to date for obtaining sufficient force electromechanically, within a practically sized package, have been dependent on the use of samarium cobalt as a high energy magnetic material. In a recently developed process, powered samarium cobalt is mechanically compressed forming a material of superior magnetic strength with high resistance to de-magnetization. The material has considerable potential for aircraft application since it is expected to ultimately reduce the size of electric motors, generators, and other transducer types as well.

Unique to the direct drive approach is the requirement that the electrical input signals to the actuator must be of sufficient strength to provide the valve driving force without supplementary amplification. Conventionally, an EHV in each channel converts a low strength electrical input to a low strength hydraulic signal. The hydraulic signal then drives a piston which produces a force multiplied, proportional mechanical output. A relatively high force electromechanical transducer, utilizing the samarium cobalt magnetic material, is substituted for the EHV and the hydraulic piston in the direct drive case. Accordingly, the transducer force output itself is applied in the same manner as the intermediate (secondary) hydraulic actuator output. The force which may thus be obtained within workable size and weight limitations appears to be adequate for most applications. Single or multiple transducers arrangements are practical to assure adequate force and redundancy. Each transducer may also have up to four individual coils. Two transducers driving a valve in a push-pull mode may be most attractive mechanically - provided that the force-fight problem may be effectively overcome. The ultimate success of this concept with respect to performance is yet to be determined; on the other hand the potential to simplify the actuator, reducing the requirement for precision hydromechanical components and very favorably affect its cost is encouraging.

Two additional direct drive advantages are that driver/valve modules could be developed to suit different requirements and the higher electrical signal power levels would reduce susceptibility to electromagnetic interference. Initial direct drive development has been accomplished in a series of programs sponsored by the U.S. Navy. An actuator utilizing dual high output torquemotors driving the control valve in a push-pull mode has been implemented and tested in the laboratory. (Ref 4) Development of a flight-worthy package to be flight tested soon is now in progress. The U.S. Air Force is now actively pursuing two direct drive approaches similar to the above. One utilizes a single linear motor drive, the other dual voice coil drivers operating in a push-pull mode. In both, the drivers are assembled directly to the valve concentric to the spool centerline. Flight test of both mechanizations being developed in the current U.S. Air Force programs is planned for the near future.

#### Power-By-Wire (PBW);

There are two basic PBW approaches now being explored by the U.S. Air Force and industry which are both dependent on electrical power to drive the control surface. The first is the Integrated Actuator Package (IAP) and the second is the Electro Mechanical (EM) actuator.



The IAP is a concept originally pursued by the U.S. Air Force primarily to improve the survivability aspects of military aircraft, particularly fighters. The package contains an essentially conventional actuator, but in addition incorporates its own modularized hydraulic supply to eliminate dependence on the central hydraulic source. The need for vulnerable hydraulic lines is eliminated, routing of electrical cables may be accomplished with comparative ease, leakage problems are eliminated, interfacing is reduced and the electrical power distribution system may be protected by circuit breakers.

The most promising IAP mechanization utilizes an electrically controlled servopump supplied by an integral reservoir. The servopump consists of an electric motor driving two pumps on a common shaft. A secondary pump maintains actuator stiffness while the primary pump provides power on demand to drive the control surface. Secondary pump displacement may be fixed or controlled as a function of primary pump output. The primary pump swashplate is controlled directly from a FBW signal utilizing an EHV, or an electro-mechanical transducer could be substituted. Control is thus quite convenient, and since mechanized as a demand system, power consumption is approximately 1/3 less than that of the conventional actuator. Power modulation of the unit is via destroking of the pump pistons as a function of swashplate angle, which must also be capable of overcenter operation to reverse flow direction to the actuator. Since the power is not valve modulated, the inherent valve pressure drop and heat rejection are avoided. In comparison to the standard secondary actuator this approach, like the direct drive, is also considerably more straightforward and less expensive.

The EM actuator is also dependent on the samarium cobalt magnetic material to develop high output torques within a reasonably sized package. The U.S. Air Force is presently sponsoring a program to develop and implement equipment for test and demonstration purposes.

There are two basic differences between the IAP and the EM actuator. First, the EM actuator utilizes no hydraulics and is powered by two brushless DC motors having samarium cobalt rotors. Secondly, electrical power to the actuator is modulated by a controller upstream which integrates the electrical signal and the primary power inputs. The two motors are mounted in-line axially at opposite ends of the package, driving the surface through a centrally located planetary gearbox. The result is a cylindrical shape of relatively small cross section which is installed and functions in a manner similar to that of a power hinge. The complete EM drive includes the actuator and the remote controller which contains a transistorized commutation circuit. Like the IAP, this concept will also be conservative of energy since it also employs an on demand mode of operation. The unit being developed will deliver up to three horsepower at the surface and is intended for primary control application. The potential of this concept is good as an alternative to hydraulically powered surfaces on high density aircraft.

#### Conclusions:

This chapter has attempted to touch upon presently significant considerations involved in the design of FCS and actuators in particular. Because so many factors are involved individual treatments have been necessarily brief although the implications should be apparent. Technically, the state-of-the-art at its current level of refinement can satisfy performance demands and nearly any other reasonable technical goals; however, a need for marked improvement definitely exists in several specific areas.

Stringent and sophisticated redundant techniques are needed to provide the reliability levels essential to the advanced actuation systems of modern military aircraft. Reduced actuator life cycle costs are needed to reduce total weapon system costs. Thus providing better assurance that future aircraft procurement programs will not be subject to significant cuts or complete elimination as a direct result of excessive costs. Much better actuator maintainability is required to improve aircraft availability and contribute to lower support costs. These objectives presently should be of primary importance and are inherent to achievement of the full potential of advanced technology and FBW in particular.

FBW and electrical control techniques are proving to be a catalyst to broaden controls applications and the functions which may be performed. These advanced techniques are the key to removing longstanding constraints, optimizing basic aircraft configuration and making new modes of control practical. Exceptionally high performance levels are possible, control is more precise and automatic functions which extend to other major systems are now realistic design options.

With this outstanding prospect, actual realization of the benefits now within reach is dependent primarily upon how well the technology is implemented; this will require reliable, cost effective systems and equipment which are practical to maintain. It is suggested that a particularly good place to start is in the design of improved FCS actuators.

#### REFERENCES

1. P. E. Blatt, "Flight Control System Advances For Near-Future Military Aircraft", Paper for SAE A-6 Committee Meeting 24 Oct 1973.
2. L. R. Appleford, Beattie, et-al, "Test & Analysis of a Quadruple Redundant Horizontal Stabilizer Actuation System", Report No. FAA-SS-72-70, April 1972.



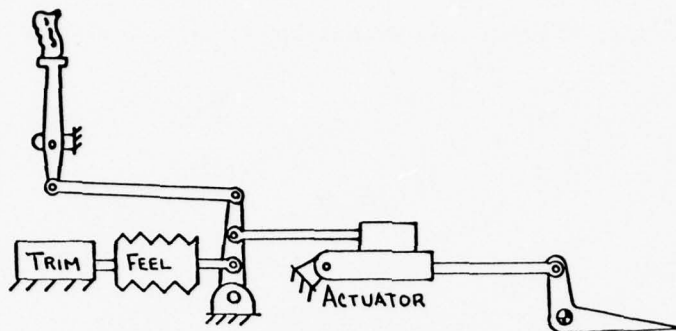


FIG. 3  
FULLY POWERED IRREVERSIBLE CONTROL SYSTEM

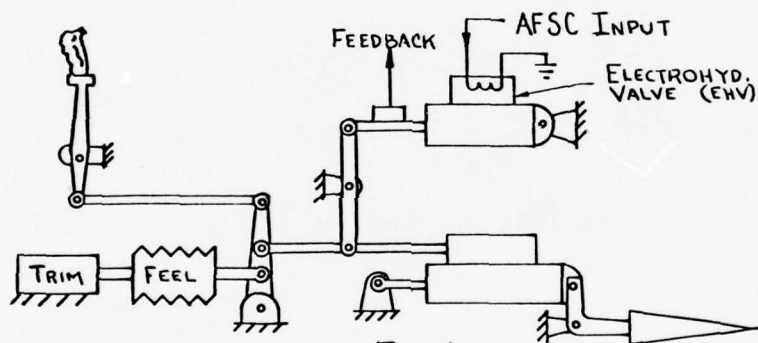


FIG. 4  
PARALLEL INPUT SERVO

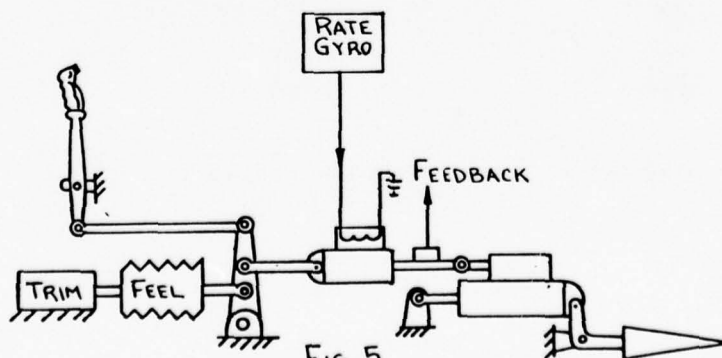


FIG. 5  
SERIES INPUT SERVO

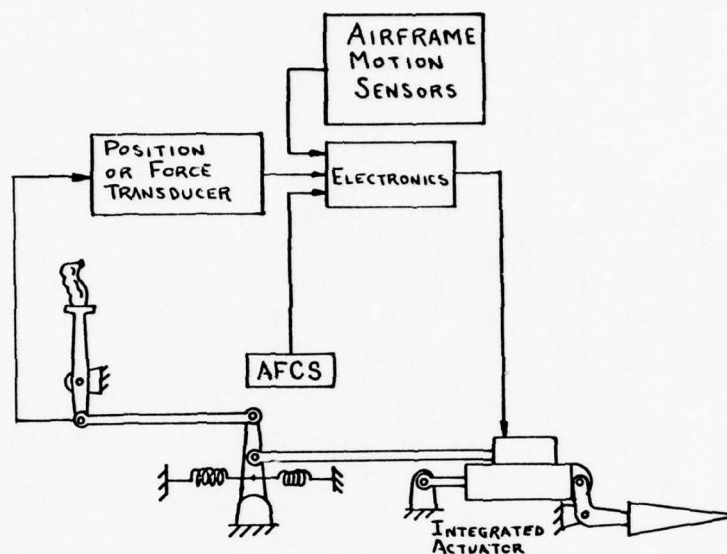


FIG. 6  
CONTROL AUGMENTATION

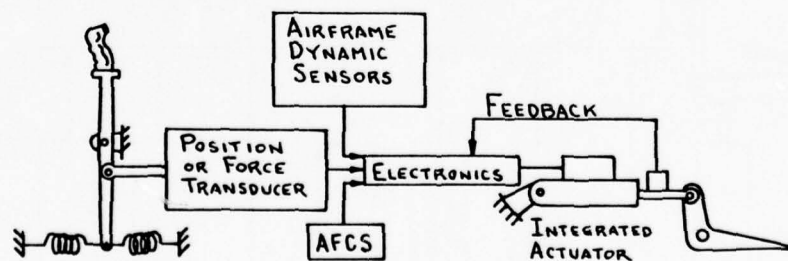


FIG. 7  
FLY-BY-WIRE CONTROL SYSTEM



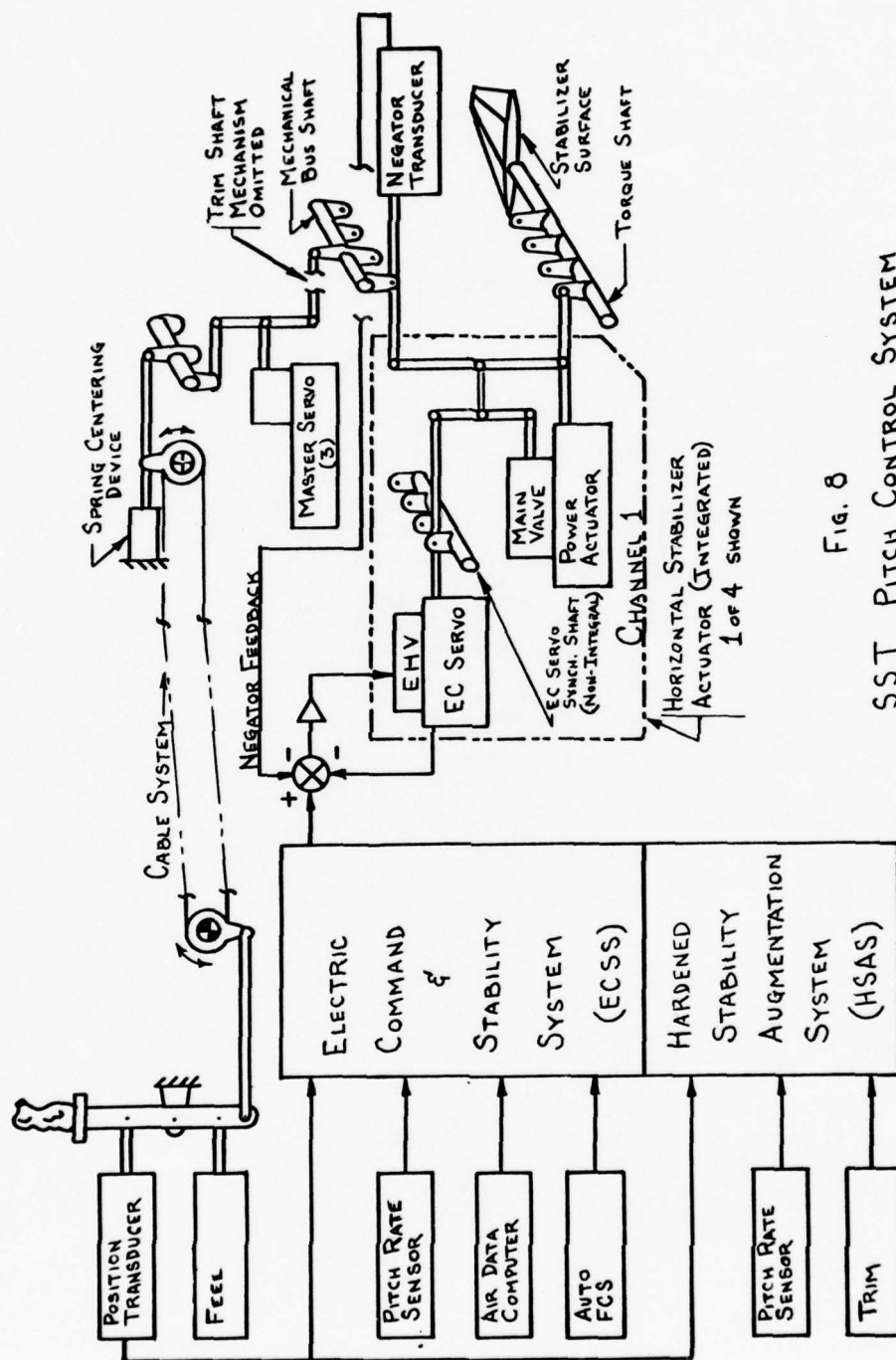


Fig. 8

SST PITCH CONTROL SYSTEM

## F-16 FLIGHT CONTROL SYSTEM DEVELOPMENT

DAVID L. CARLETON  
AERONAUTICAL SYSTEMS DIVISION  
WRIGHT-PATTERSON AFB, OHIO 45433

## SUMMARY

The purpose of this paper is to outline development procedures required to implement the fly-by-wire flight control system in the F-16 aircraft. Several developmental efforts were required to implement the flight control system into the aircraft. These efforts include the design and development of specific hardware, including the sensors, actuators, and the flight control computer itself. Once the subsystems were developed, the process of integration and definition of the flight control system became a developmental effort. Once the hardware was integrated into the aircraft, the developmental effort then swung towards on-aircraft tests to ensure that the flight control system was compatible with the airframe within the operational flight envelope. Once these ground tests were completed, the development effort then concentrated on the flight test portion of the program where the flight control system was optimized precision tracking in the air superiority and ground attack role. The F-16 flight control system development was rather unique inasmuch as it was a two-fold effort. A development effort was undertaken to ensure that the prototype aircraft could indeed meet the safety of flight requirements and then the effort swung towards full scale development of the fly-by-wire flight control system for a production aircraft application. The process is illustrated in Figure 1, which expands upon the development process and shows how the various steps interact. It also demonstrates that iterative nature of the process.

## SYSTEM DESCRIPTION

In order to understand the procedures required to develop a fly-by-wire flight control system, a brief description of the system is necessary. The flight control system in the F-16 is a fly-by-wire system in all three control axis. Each control axis has four separate electrical channels and two hydraulic systems for redundancy purposes. There are no mechanical backup controls in the flight control system. Each control axis has pilot input commands, (i.e., pitch and roll stick force or rudder pedal force) and each control axis employs active aircraft feedback so that the flight control system is an aircraft rate and acceleration command system as opposed to a more conventional control surface position command system. During subsonic flight the aircraft is statically unstable in the pitch axis so this feedback or command augmentation system is required in order to fly the aircraft. In addition to rate and acceleration sensors to control the aircraft, the flight control system receives inputs from the central air data computer to schedule flight control system gains as a function of flight condition. In order to ensure full time operation of the flight control system, there are seven electrical power supply sources that the flight control system can call on to power its four channels of electronics in each axis. Additionally, there are two hydraulic systems in the aircraft with a backup emergency hydraulic pump in the event of an engine failure to provide hydraulic power to the flight control systems. Block diagrams of one channel of the longitudinal, lateral and directional portions of the flight control system are shown in Figure 2 for the YF-16 and Figure 3 for the F-16. Reference 1 contains a complete description of the system. As shown in the Figures, extensive use is made of non-linear shaping and flight condition gain scheduling within the flight control system.

## HARDWARE DESIGN AND DEVELOPMENT

The first portion of the development of the system was to define the configuration of the hardware to be used in the flight control system, or the design process. A great deal of material has been written about the system design philosophy. The key design points will be touched upon as they relate to the development process. Because this was the first fly-by-wire flight control system employed in a production aircraft, the flight control system remains separated from all other subsystems within the aircraft. The flight control computer is a four channel, three axis analog computer which represents an extension of current high gain, high authority control augmentation system technology. The computer that was built and developed for the prototype aircraft was a 32 board analog computer employing integrated circuits consisting of approximately 5500 piece parts. The unit was designed to be forced air cooled although it can operate for extended periods of time without cooling air before failing. It was subjected to the standard qualification tests to ensure structural integrity within the unit. The quad redundant computation system consists of three active channels and a monitored channel. This represented an extension of the logic in the F-111 Control Augmentation System. The design is such that at each voting point within the computer, a mid-channel is selected as being the "good signal". In the event of a channel failure, the monitor channel then switches in to become the third channel and the mid-channel selection process continues. In the event of a similar failure at the same point of the computer, the remaining two channels operate with the value closest to a null condition being selected as the good channel. A rather interesting scheme of unit signal selection is employed at each voting point in the system. The signals downstream of the voter are compared with the upstream signal and then processed by an op-amp. The high and low signals drive the amps into saturation and the median signal only is processed through the voter. The gyroscopes and accelerometers used in the flight control system are also current state-of-the-art gyroscopes. These sensors are testable with torquing signals. The prototype program used the servo valves and actuators from the F-111. This design was changed in the production aircraft to include integrated servoactuators with the power control cylinder. All servo-actuators in the aircraft are identical and the power actuators are identical with the exception of the rudder actuator. Once again these sensors and actuators were also subjected to bench qualification tests to ensure the fact that they would indeed operate reliably in the aircraft. Once the hardware design and development process has started, the second development phase began.

#### SYSTEM DEFINITION AND INTEGRATION

This phase dealt with system integration and system definition. This phase made extensive use of simulation facilities to accomplish two objectives. First objective was to define the control logic within the flight control system using an engineering simulation facility, and the second objective was to integrate the hardware in the simulator facility prior to installing it on the aircraft to ensure compatibility. The control law simulation was conducted on a fixed base simulator to define and verify the control logic for use in the aircraft. A small visual facility was made available for the purpose of evaluating handling qualities during formation and gunnery tasks in the simulator. The process made use of stability and control derivatives that were gathered from the wind tunnel. Several advanced control features were defined and developed in the simulator to enhance handling characteristics. The angle-of-attack limiter was defined and developed in the fixed base simulator. The yaw axis sideslip rate feedback was defined. These two functions required concurrent development because a controlled flight boundary exists in terms of angle-of-attack and sideslip. The simulator is the ideal tool for this definition process. The time constant in the pitch rate feedback loop and the aileron-rudder interconnect logic are additional parameters that ideal candidates for simulator development. In addition to refining the handling qualities in the operational flight envelope, stall/spin characteristics and spin recovery techniques were also investigated on the simulator. The second purpose of this simulation was also to integrate flight hardware into the control law simulator to ensure that the fact that it did operate as prescribed. This also represented a further refinement of the simulation for it included the effects of the actual hardware as it would perform in the aircraft. Selected failure modes were also inserted and the built-in-test logic was validated on this phase of the program. This phase of the program replaced the requirement for an "iron-bird" type flight control mockup simulation. These "iron-bird" tests were conducted concurrently with the handling qualities development tests.

#### AIRCRAFT GROUND TESTS

The third series of development tests were conducted on the flight control system installed in the airframe. These development ground tests consisted of the standard ground tests conducted with a closed loop flight control system. Limit cycle, structural resonance, frequency response tests, and also a series of electromagnetic interference, or lightning tests were conducted on the flight control system. Because the electrically implemented fly-by-wire flight control system contains no direct mechanical linkage from the control stick to the flight control surface, immunity from the effects of lightning strikes and other electromagnetic interference was considered to be a major demonstration requirement to ensure safety of flight. Several tests were conducted by increasing the duration and amperage of simulated lightning strokes on the actual aircraft and measuring the resultant voltages at selected points in the flight control system. Two series of tests were run. The first test used a maximum current of 260 amps and extrapolated the data to a maximum strike of 200,000 amps. The maximum induced voltage observed during this test ranged from .3 to 1.8 volts. A second series of tests were conducted using a peak current of 3000 amps. The results of these tests were that maximum transient induced voltages ranging from .4 to 13.8 volts were measured. The duration of these voltages were sufficiently short such that no interference with the system operation would be anticipated with a lightning strike.

Additional compatibility tests that were performed were limit cycle and structural resonance tests. The purpose of conducting the structural resonance tests were to ensure that no structural coupling between the airframe and system existed. The purpose of conducting the limit cycle tests were to ensure adequate phase and gain margin were maintained within the flight control system so that the system did not go unstable. As a result of these tests, some modifications were made to the structural filtering in the flight control system. The requirement to maintain stability margins with the large number of stores to be carried by the production aircraft necessitated several pitch axis modifications, such as altering the normal acceleration lag, reducing pitch rate gain, reshaping the dynamic pressure scheduled gain and modifying the lead-lag networks. A structural mode was encountered in flight that was not discovered during the ground tests. This was a coupling of a pitch mode in a wing tip mounted missile with the lateral portion of the flight control system to produce a asymmetric bending mode. This mode varied in frequency and damping with missiles on or off. A notch filter in the lateral axis was modified to include the frequency range encountered in flight, and the problem was eliminated. Once these ground tests were successfully completed the aircraft was then certified as being safe for initial development tests flights.

#### AIRCRAFT FLIGHT TESTS

Two phases of testing were accomplished during the flight test program (Ref 2) to ensure that the handling qualities were optimized for the air-to-air and air-to-ground role and that the flight control system did not produce any adverse effects during high angle of attack maneuvering or degrade spin recovery capability in the out of control flight regime. Before discussing these it is rather interesting to note that the first change during the development process that was made to the flight control system was that of reducing lateral sensitivity with the landing gear down due to an inadvertent first flight that took place during a high speed taxi test. Here it was determined that the roll command gains, or lateral sensitivity was too high and that during the test the pilot actually was in a position where he was forced to take the aircraft off to avoid running off the side of runway. After this flight, the roll command gain was halved with the gear down. The development of the flight control system then proceeded through the flight test by optimizing the handling qualities of the vehicle using the air-to-air tracking test technique or handling qualities during tracking. In addition to gathering stability and control data, 26 data flights were required to optimize the system for air-to-air tracking. This was conducted by using six pilots to provide handling qualities data that gave both qualitative and quantitative data on the aircraft tracking capabilities. During the course of this program, 60 configuration changes were made to the flight control system to enhance the aircraft's capabilities. A summary of the major changes made to the flight control system are listed in Table 1. The major changes that were accomplished included modification of the roll notch filter and roll feedback gains, the pitch rate feedback washout time constant, the C-star blend ratio, and stick force breakouts. Several evaluation flights were also conducted with various limited displacements on the aide arm controller. A synopsis of the tracking optimization process is shown in Figures 4 and 5.



These plots compare the F-16 with other current aircraft and test programs to show that the side arm controller is capable of demonstrating excellent tracking capability.

Once the system was optimized for flight within the controlled flight envelope a high angle of attack investigation was undertaken to ensure that the flight control system did not contribute adversely to the spin recovery characteristics of the aircraft. Although the flight control system does contain an angle of attack limiter, the angle of attack limit was exceeded on several occasions during the flight test program. These out-of-control situations occurred at very low airspeeds accompanied with high angular rates or when abnormal control inputs were used to maneuver the aircraft. The aircraft did enter a spin while the pilot attempted to roll the aircraft with rudder only. Recovery was accomplished with pilot inputs. As a result of exceeding these limitations, several modifications to the basic flight control system were proposed. The first one was to schedule pitch acceleration feedback with increasing angle of attack to enhance the controllability at low airspeeds. The second modification that was recommended was to reduce the amount of commanded roll rate as airspeed decreased to reduce the magnitude of the inertial coupling term.

#### PRESENT STATUS

The prototype (YF-16) program has been completed. One of the prototype aircraft has been fitted with vertical canards and is currently under test as a control configured vehicle. The other prototype aircraft is conducting follow-on avionics testing in support of the F-16 Full Scale Development (FSD) program.

The first FSD aircraft is about to be rolled out and has successfully completed ground testing.

#### CONCLUSIONS

The procedures used to develop the F-16 fly-by-wire flight control systems are less time consuming and costly and allow for more design flexibility as compared to those procedures required to develop a mechanical augmented "conventional" system. Hardware and control logic can be developed concurrently, hereby reducing the complexity of hardware simulations and eliminating the requirement for an "iron-bird". The design benefits of full authority fly-by-wire to exploit active control technology are obvious. A developmental benefit is that afforded by the ease by which changes can be implemented via ground or in-flight system reconfiguration. This was demonstrated in the flight test program. It is anticipated that the production aircraft development cycle can be completed as effectively as the prototype program was.

#### REFERENCES

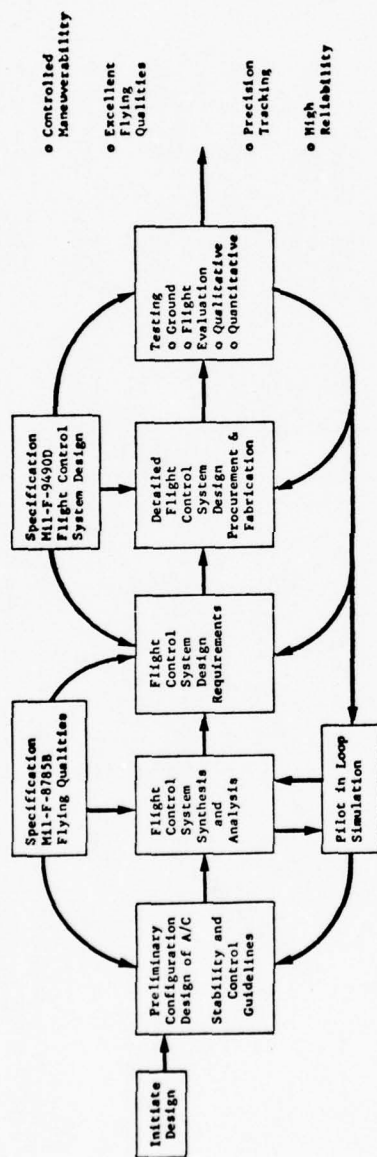
1. F-16 Flight Control Design Analysis Report, G.D. 16PR276, 13 July 1976, General Dynamics Fort Worth Division.
2. Eggers, James A., Major, USAF, Bryant, William F., Jr., Major, USAF, Flying Qualities Evaluation of the YF-16 Prototype Lightweight Fighter, AFFTC-TR-75-15, Air Force Flight Test Center, Edwards AFB, California, July 1975.



TABLE 1

CONFIGURATION CHANGES

1. Modified roll rate command to allow 0.5 gain selection for reduced lateral sensitivity.
2. Rewired FCS to permit speed brake operation in flight with landing gear extended for better SFO pattern control.
3. Rewired trim panel to produce 50 percent less roll trim rate to reduce roll trim sensitivity.
4. Increased rudder feel spring breakout force from 10 lb to 22 lb to reduce sensitivity.
5. Modified FCS to correct FCS/structural coupling and pitch sensitivity during tracking. Changes included a roll notch filter, revised roll gains, and an increase of the pitch rate wash-out time constant from 1.0 to 2.25 seconds.
6. Pitch rate wash-out time constant lowered from 2.25 to 1.5 seconds. A lead term in leading edge flaps commanded was added to improve performance.
7. Installed 300-degree-per-second roll rate gyro to improve FCS lateral axis.
8. Reworked FCS panel to improve FCS self-test and changed the pitch rate wash-out time constant from 1.5 to 1.0.
9. Decreased  $n_z/\dot{\theta}$  gain from 5:1 to 2.5:1 and lowered alpha limiter from 28 to 26.7 degrees.
10. Stiffened the side stick by increasing roll force limits to  $\pm 15$  pounds and increasing pitch output to 30 pounds.
11. Changed pitch sensitivity by adjusting  $F_e$  prefilter time constant from 1/4 to 1/2.27.<sup>e</sup>
12. Replaced fixed stick controller with linear pitch displacement stick.
13. Installed linear pitch displacement stick.
14. Removed displacement stick and installed force stick controller.
15. Eight-radian lag installed in pitch axis to reduce high-gain task pitch sensitivity.
16. Four-radian lag reinstalled in pitch axis.
17. Revised displacement stick to reduce travel by 50 percent.
18. Installed FCS computer with 8-radian lag in pitch channel.
19. 3.6-radian lag in roll filter installed in FCS.
20. Eight-radian lag in pitch channel reinstalled.



STABILITY AND FLIGHT CONTROL DESIGN PROCESS

Figure 1

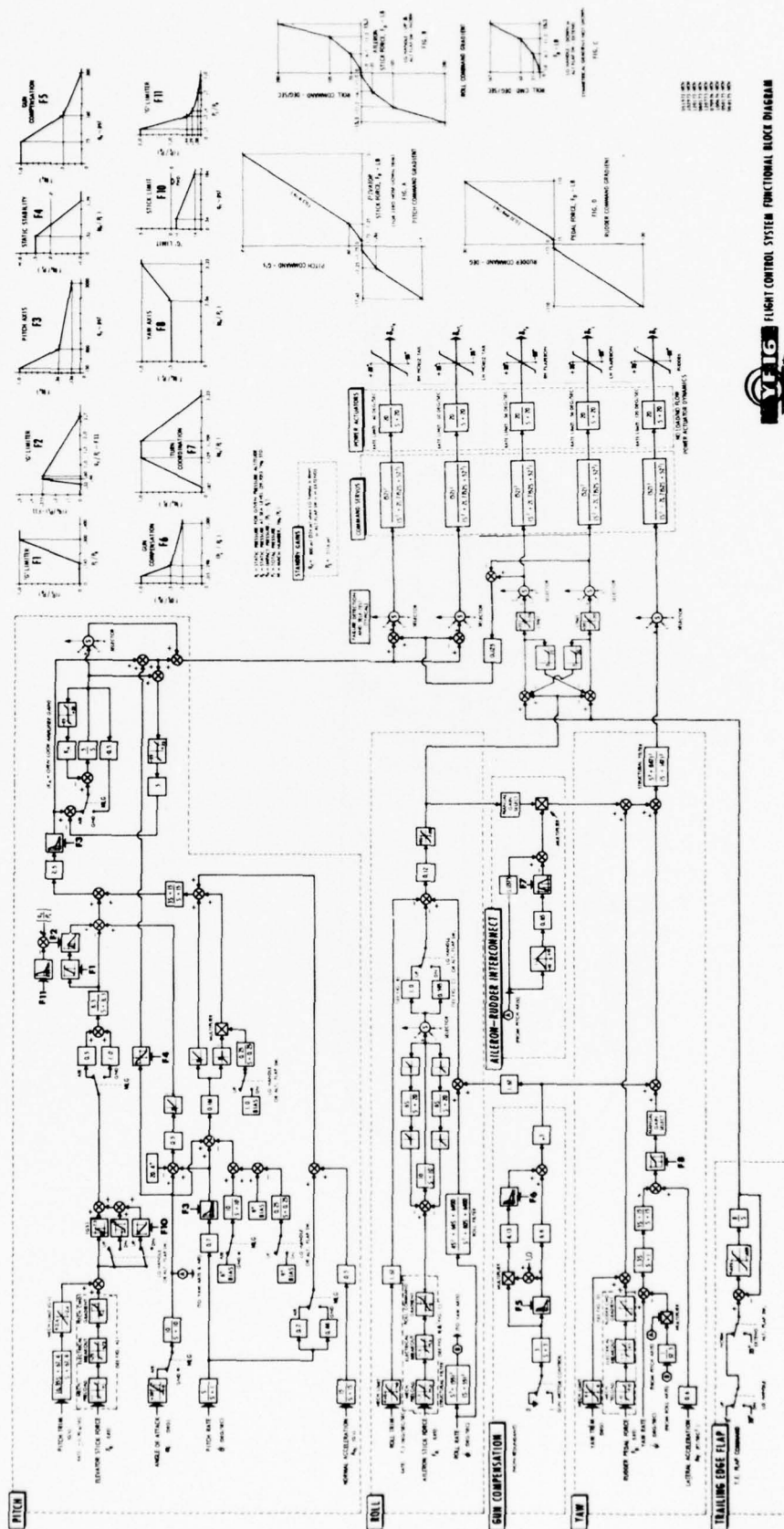


Figure 2

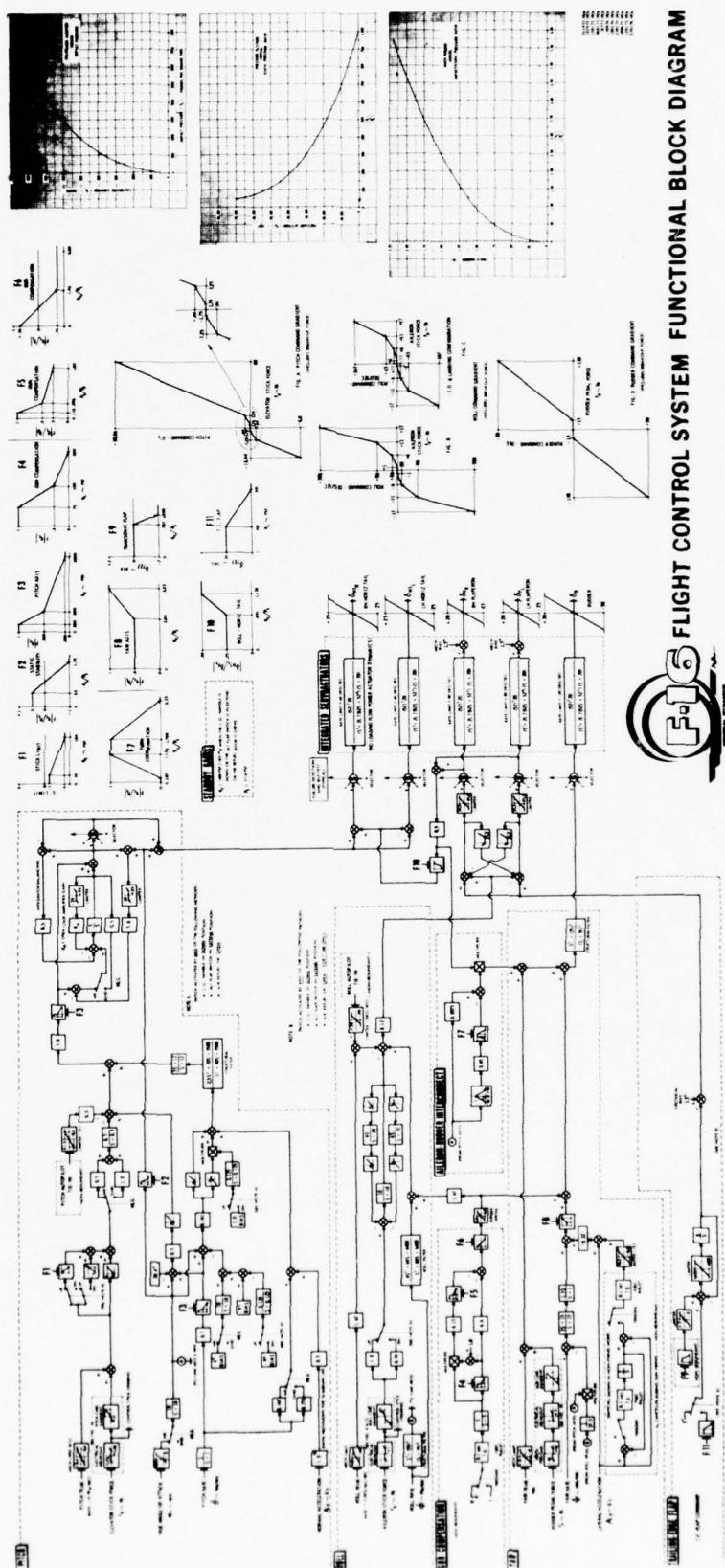


Figure 3



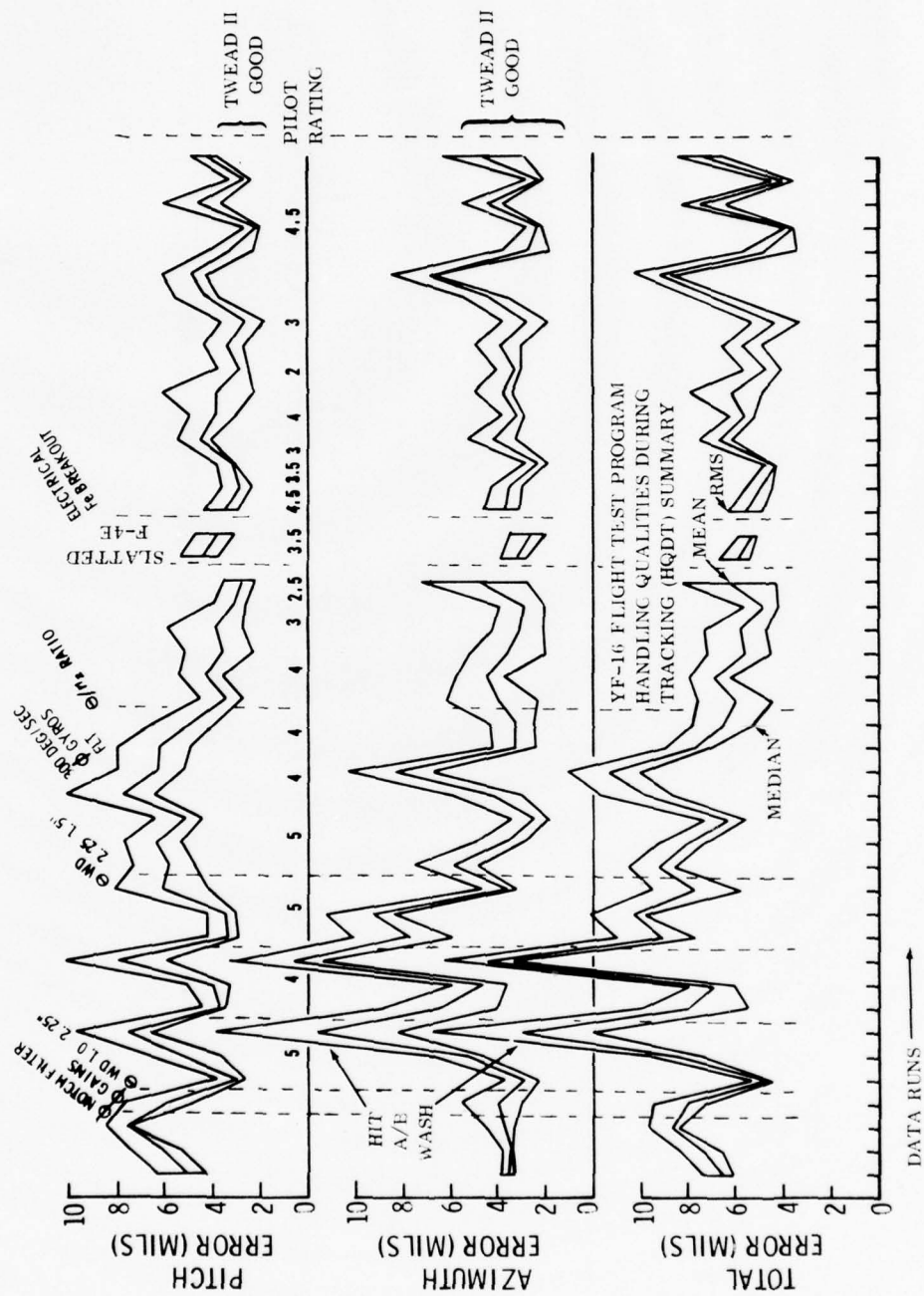


Figure 4

## YF-16 HQDT PROGRAM SUMMARY

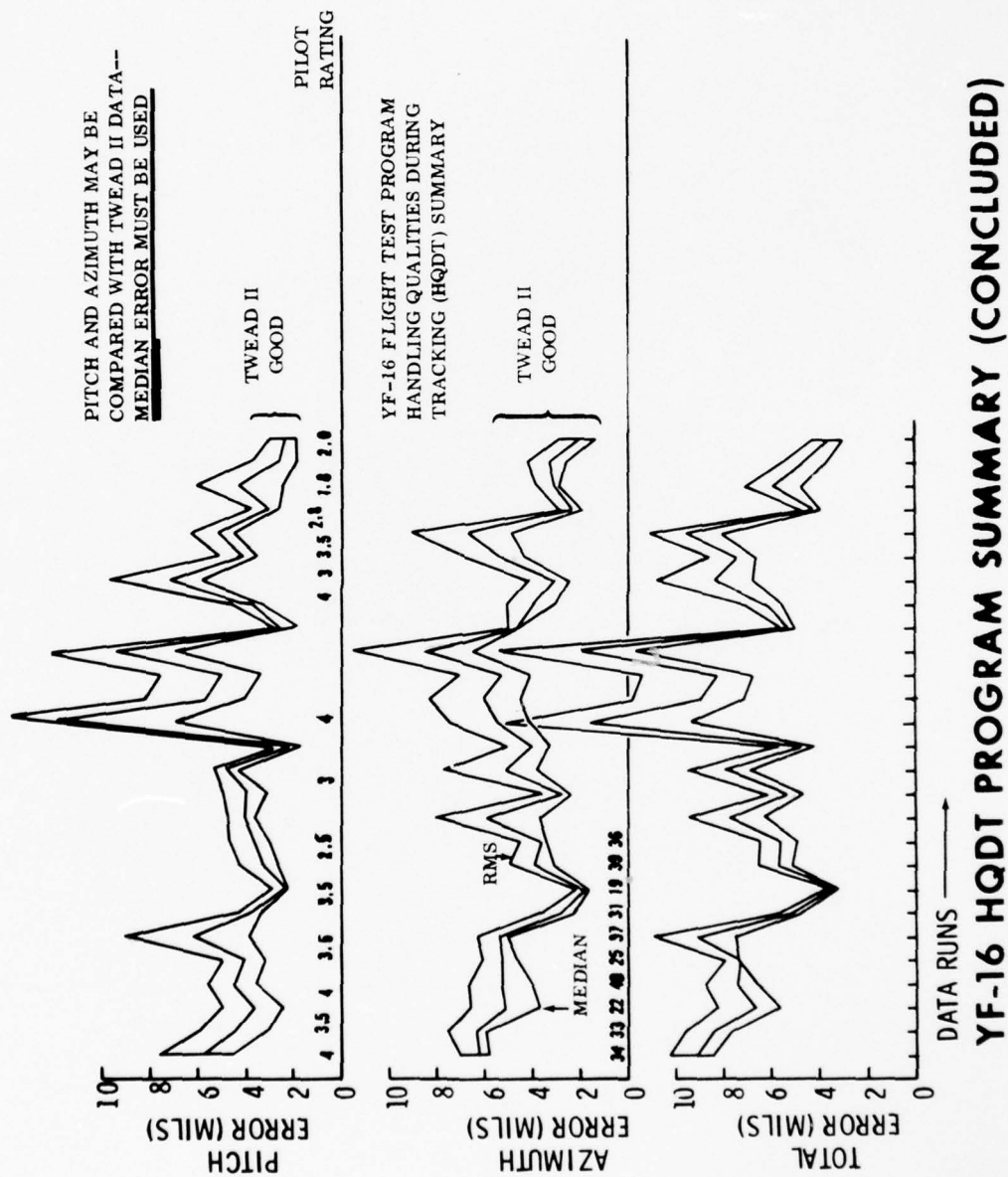


Figure 5

# JA-37 DIGITAL AUTOMATIC FLIGHT CONTROL SYSTEM (DAFCS) SELF-TEST DEVELOPMENT

by

D. G. Bailey

Government and Aeronautical Products Division  
Honeywell Inc.  
1625 Zarthan Ave.

St. Louis Park, Minn. 55416

and

Kjell Folkesson

SAAB SCANIA  
Linköping, Sweden

## SUMMARY

The SAAB-SCANIA JA-37 Viggen Interceptor, Figure 1, which enters service with the Swedish Air Force in 1978, will be equipped with the world's first production Digital Automatic Flight Control System (DAFCS). The DAFCS was designed and developed by Honeywell's Government and Aeronautical Products Division under contract with Sweden's Air Materiel Department (FMV). SAAB-SCANIA was largely responsible for development of detailed system requirements and integration of the DAFCS with other aircraft systems.

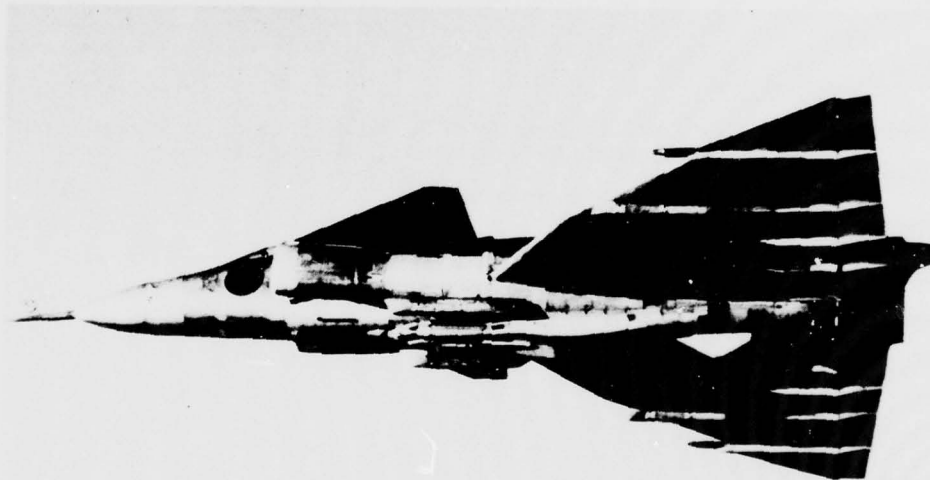


Figure 1. SAAB-SCANIA JA-37 Viggen Interceptor

The DAFCS is currently in the final stages of flight test development, with series production hardware deliveries scheduled to begin in 1977. Following is a description of DAFCS self-test development, which produced flight line and in-flight self-test capability consistent with demanding flight safety and built-in test effectiveness requirements.

Self-test development included verification of self-test effectiveness to a degree sufficient to --

- Support the decision that a single processor DAFCS mechanization could provide adequate in-flight fail safety, and that mechanization of a redundant second processor was unnecessary in the production JA-37 DAFCS, and to --
- Confirm that the DAFCS preflight self test was in compliance with maintenance performance test/fault localization requirements.

Adequate in-flight fail safety was verified by demonstrating that the in-flight monitoring function was extremely effective in detecting "potentially catastrophic" failure modes. ("Potentially catastrophic" failure modes are those that produce transients in excess of specified limits.) Effectiveness in detecting potentially catastrophic failure modes was demonstrated to be well above 99 percent. Overall fault detection, including potentially catastrophic as well as non-catastrophic failure modes, was closer to 99 percent. The effectiveness of the in-flight monitoring function was demonstrated to be compatible with the allowable  $1 \times 10^{-6}$  catastrophic failure probability for a 90-minute mission.

The test program that demonstrated these results is described in the following paragraphs.

## ABBREVIATIONS

DAFCS	- Digital Automatic Flight Control System
FMV	- Swedish Air Materiel Department
CAS	- Control Augmentation System
CPU	- Central Processor Unit: The Honeywell HDC-301, a single-card digital processor
WDT	- Watchdog Timer
LSIC	- Large-Scale Integrated Circuit
I/O	- Input/Output
A/D	- Analog to Digital
D/A	- Digital to Analog
DCM	- Dynamic Computation Monitor
FMEA	- Failure Modes and Effects Analysis
FMET	- Failure Modes and Effects Testing
P <sub>CF</sub>	- Probability of Catastrophic Failure
BIT	- Built-In Test
LRU	- Line Replaceable Unit

## DAFCS DESCRIPTION

DAFCS Functional Configuration

The Digital Automatic Flight Control System (DAFCS), Figure 2, is a high-authority, fail-safe, single processor digital flight control system that provides the following functions:

- Control Augmentation System (CAS), Normal Mode
  - Stability Augmentation
  - Transonic Trim Change Compensation
- Control Augmentation System, Aiming Mode
- Attitude Hold
  - Pitch Attitude Hold
  - Roll Attitude Hold
  - Heading Hold
  - Control Stick Steering
- Altitude Hold
- Automatic Airspeed Control

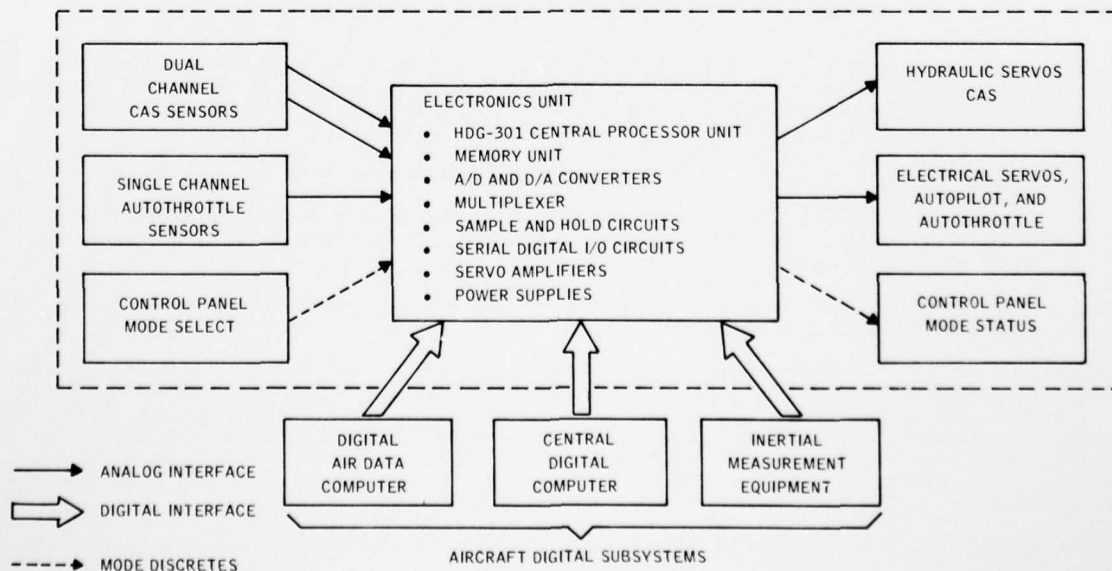


Figure 2. JA-37 DAFCS Mechanization



The DAFCS is mechanized with inputs from dual analog sensors for the control augmentation function, and with single serial digital reference signals for attitude control modes. The single HDC-301 digital central processor unit (CPU) performs control law computation, monitoring, and test functions, and outputs command signals to the single control surface servos.

#### DAFCS Self-Test Configuration

The JA-37 DAFCS self test is designed to provide in-flight fail safety as well as performance test/fault localization to facilitate system maintenance. As shown in Figure 3, the DAFCS self test is an integral part of the JA-37 system test concept, which includes:

- In-flight Monitoring
- Flight Line Test
  - Power-Up Test/First Line Check
  - Internal Performance Test/Internal Fault Localization
- Depot Level Test
  - Performance Test
  - Fault Localization to Shop Replaceable Unit

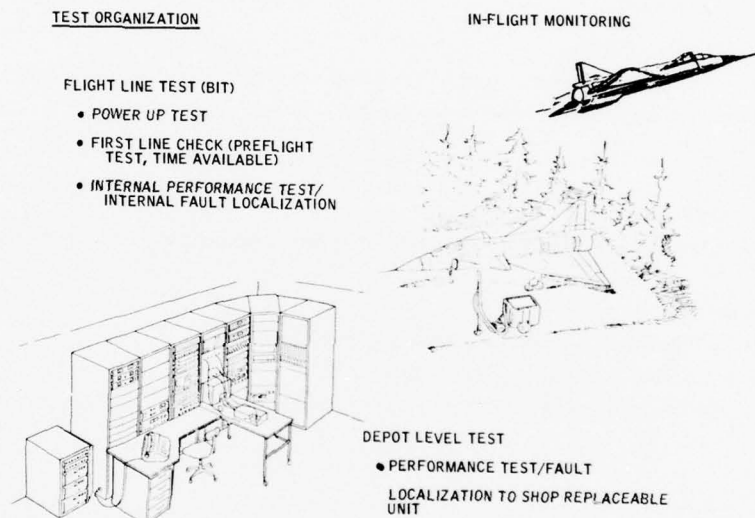


Figure 3. JA-37 DAFCS Self-Test Organization

DAFCS in-flight monitoring and flight line testing are accomplished via built-in tests performed by the DAFCS digital computer. For depot level testing, built-in test is augmented by external automatic test equipment for fault localization to shop replaceable unit (e.g., printed circuit board).

#### DAFCS Self-Test Requirements -- DAFCS self-test requirements can be summarized as follows:

- In-flight Monitoring, Fault Detection - Probability of undetected failure resulting in "potentially catastrophic" failure shall be less than  $1 \times 10^{-6}$  during a 90-minute mission. "Potentially catastrophic" failure is defined as:
  - Normal Acceleration,  $|\Delta N_z| > 2 \text{ g}$  incremental
  - Lateral Acceleration,  $|\Delta N_y| > 0.5 \text{ g}$
  - Other parameters
- In-flight Monitoring Nuisance Disengage - Probability of nuisance disengage shall be less than  $1 \times 10^{-3}$  during a 90-minute mission.
- Flight Line Performance Test - One hundred percent check of in-flight monitoring and disengage mechanism required. Overall fault detection probability must be greater than 95 percent, where a "fault" results in performance outside functional requirements.
- Flight Line Fault Isolation - For detected faults, fault localized to line replaceable unit (e.g., a black box) with 95 percent confidence.
- Depot Level Test - ATE requirements only.

**DAFCS Self-Test Mechanization, In-flight Monitoring** -- DAFCS in-flight monitoring, Figure 4, must be fast and comprehensive to meet requirements because the DAFCS authority is potentially destructive; i.e., authority in the low-altitude, high-speed flight regime is approximately 10 g. Hard-over failure will produce transients in excess of requirements if left undetected for more than 0.1 second at high-speed flight conditions. DAFCS in-flight monitoring can be summarized as follows:

- **Sensor Monitoring**
  - Comparison monitoring of feedback signals from dual-redundant inner-loop CAS sensors (rate gyro packages, normal and lateral accelerometer packages, stick force and stick position transducers).
  - Parity and format checks on digital inputs from nonredundant outer loop sensors, and software limiting of outer loop sensor signals.
  - Monitoring of critical gains using an independent air data source to protect against air data computer failures.
- **CAS Servo Monitoring**
  - Comparison of actual performance against digital models.
- **Input Circuitry Check**
  - Accomplished by software comparison of A/D converted power supply references to known constants.
- **HDC-301 CPU Monitoring**
  - Self test - Exercises all instructions and control critical to flight safety.
  - Dynamic Computation Monitoring (DCM) - A continuous check of critical HDC-301 CPU and I/O functions by CPU dynamic control of an external independent analog element. CPU failures affecting CPU dynamic computation capability will trip this monitor.
  - Watchdog Timer (WDT) - Verifies that the computation cycle is completed within a specified time limit.
  - Real Time Clock Monitor - Assures that the computation period has not exceeded a nominal value.
- **Memory Monitoring**
  - Parity checks on all words accessed from memory.
  - Periodic sum checks of critical instruction, constant, and scratchpad locations.
  - Continuity monitoring to assure proper program flow through critical instructions.

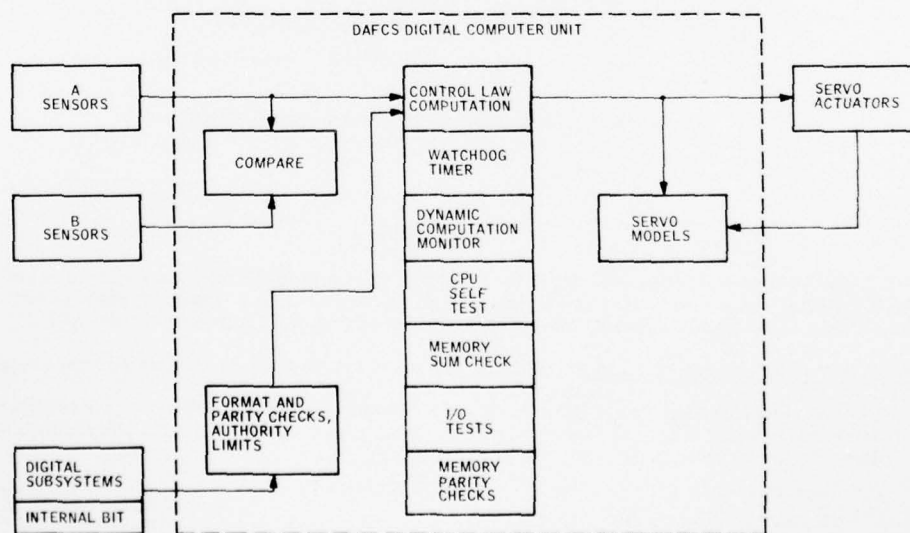


Figure 4. JA-37 DAFCS In-Flight Monitoring

**DAFCS Self-Test Mechanization, Flight Line Testing** - The DAFCS flight line test includes the following:

- Processor Self Test
- Memory Test
  - Parity Checker Test

- Memory Sum Check
- Memory Addressing Test
- A/D, D/A Converter Test
  - Calibration test via wrap-around testing
- HDC-301 CPU Monitor Test
  - Test of Watchdog Timer for proper time-out period.
  - Simulation of CPU failure to check for proper DCM failure indication.
- Disengage Circuitry Test
  - Attempt engagement with WDT, DCM failed. Verify no engagement.
- Power Supply Reference Test
- Input/Output Signal Conditioning Test
  - Stimulate all input electronics and measure dynamic response.
  - Check output electronics via wrap-around testing.
- Sensor Tests
  - Stimulate sensors, measure dynamic response.
  - Interrogate digital sensor data valid signals.
- Servo Tests
  - Exercise all servos to measure dynamics, rate and position limits, thresholds, etc.

A power-up test verifies DAFCS in-flight monitoring capability by using a sequence of the described resident tests automatically at power on. A first line check, which is also resident in DAFCS memory, is a sequence of the described tests initiated by pilot command to determine the system's operational status. An internal performance test/internal fault localization test program employs the described tests expanded for fault localization as loaded into DAFCS memory from a portable tape unit and executed via a cockpit control panel. Measurement values and test position numbers are sent to the aircraft central computer for recording.

#### DAFCS SELF-TEST DEVELOPMENT

Following is a description of the various tasks included in the definition, analysis, and verification of DAFCS in-flight monitoring and flight line test functions, as summarized in Figure 5, JA-37 DAFCS Self-Test Development.

##### In-Flight Monitoring Development

DAFCS in-flight monitoring development was initiated prior to the formal JA-37 Development Program contract award in May 1973. Various monitoring concepts had been defined and analyzed, and a prototype Honeywell digital flight controller had been flight tested in a SAAB-SCANIA AJ-37 aircraft. Pre-award cost tradeoff studies had indicated that the DAFCS should be mechanized as a single processor digital configuration.

The contracted JA-37 DAFCS Development Program was aimed at systematic, orderly development of such a single processor digital configuration. The program included prototype and preproduction hardware development and test phases. The prototype hardware intended for flight test in a JA-37 aircraft (modified AJ-37 aircraft) was to be mechanized as a dual processor configuration. Thus the dualized prototype hardware could be undergoing functional flight test development while the critical single processor monitoring functions were developed and verified in preparation for single processor preproduction system flight test in aircraft JA-37-8 (first JA-37 preproduction air frame). Preproduction hardware was designed to allow optional dual CPU mechanization, but this option was not exercised as the safety of the single processor configuration was satisfactorily verified.

The actual development program schedule is presented in Figure 5. Key decision dates in this in-flight monitoring development program included:

- JA-37-21 First Flight Approval, June 4, 1974 - Decision was made that the flight test prototype safety configuration, with dual processors, had been sufficiently verified by analysis and testing to allow the 37-21 DAFCS to fly.
- Safety Configuration Review, November 21, 1974 - Safety specification compliance was established. This was based on the calculations of catastrophic failure rate using agreed failure rate data and agreed mission time, failure transient magnitude data from simulation and flight test, and calculated nuisance disengage probability.
- JA-37-8 First Flight Approval, October 1, 1975 - Decision was made that the single processor safety configuration had been sufficiently verified by analysis, ground testing, and flight testing to allow the single processor 37-8 DAFCS to fly.

The safety configuration development involved a preliminary round of verification activity prior to prototype hardware flight test in the JA-37-21 aircraft. This activity included functional level failure mode effects analysis (FMEA), prototype hardware safety studies, and rig studies verifying the ability of the preproduction safety configuration to detect virtually all potentially catastrophic failure modes. The preproduction safety configuration verification process required:

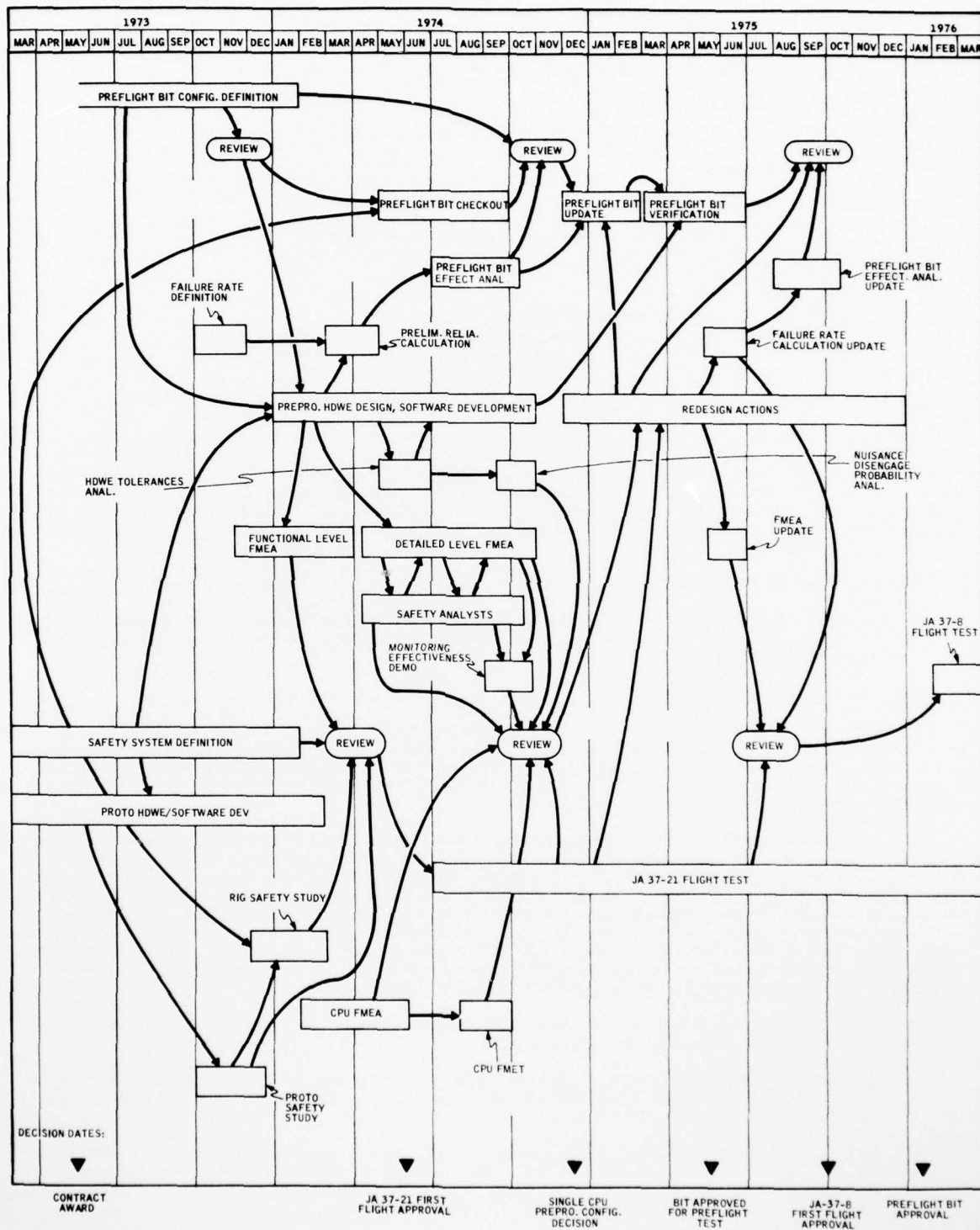


Figure 5. JA-37 DAFCS Self-Test Development



1. Analysis of integrated circuit failure modes at the gate level to characterize faults in terms of "stuck at" output pins.
2. Simulation with actual hardware of these "stuck at" failure modes where feasible, by opening or shorting integrated circuit output pins and verifying detection by DAFCS self-test software. Any remaining "stuck at" output pin faults were analyzed for detectability.
3. Documentation of effects of all failures. For undetected failures, noncatastrophic effects on aircraft were confirmed via simulation studies.

Key tasks in this safety configuration development are described in the following paragraphs.

Safety System Definition -- Monitor functions were defined to detect conceivable faults in all the DAFCS hardware components. Computation rates for monitoring functions were established based on fault reaction time requirements, and the number of fault indications required prior to fault reaction to minimize nuisance disengage probability. Monitoring of analog devices (e.g., dual sensor comparison monitoring, servo to servo model comparison monitoring) required at least three successive miscompares prior to fault reaction (disengagement). Digital hardware monitoring functions (e.g., CPU self test, memory sum check) activate fault reaction after a single fault indication.

The monitor function computation rate was set sufficiently high such that time required for fault identification (e.g., three computation cycles for analog devices) plus time required for fault reaction (e.g., recentering servo actuators) was short enough to restrict failure transients to allowable limits.

Hardware and software were developed to provide a prototype safety configuration that consisted of a single processor DAFCS mechanization with a redundant processor used to duplicate the CAS computations and provide additional safety during the early developmental flying in the 37-21 aircraft. Special emphasis was placed on making the mechanization safe without the second processor such that the second processor could be eliminated with confidence after early test flying in the 37-21 aircraft.

Functional Level FMEA -- The functional level FMEA included analysis of each functional block of the DAFCS to determine failure modes and the corresponding failure effects. Detectability of each failure mode by in-flight monitoring was determined. The functional level FMEA was the first round of analyses of monitoring configuration development. The functional level FMEA was followed by detailed level FMEA, which served to verify the fully developed system design.

The detailed level FMEA consisted of subdividing the DAFCS into 56 functional blocks and identifying all failure modes associated with each block. Of the 343 failure modes identified, 126 were determined "potentially critical", and these became the focal point for further investigation. Where there was a reasonable chance that a system safety or reliability improvement could be identified by going to a lower level or where a specific portion of the FMEA was identified as necessary for a design decision, the functional level FMEA was expanded.

Prototype Safety Studies -- Safety studies were designed to identify DAFCS failures with the potential to produce most significant aircraft transients. These failures were further evaluated during the "rig" studies, described later. Prototype hardware was employed to evaluate the ability of the in-flight monitoring functions to detect a broad set of simulated hardware failures and cause system disengagement consistent with flight safety requirements. The DAFCS unit was programmed with the flight software, which included the in-flight monitoring functions. Failures were simulated, and the in-flight monitoring reaction to these failures was observed. The following monitoring functions were examined during this study:

- Dual Element Monitoring
- I/O Monitoring
- Memory Monitoring (parity, sum check, etc)
- CPU Monitoring (self-test routine, dynamic computation monitor)
- Engage/Disengage Circuitry

Hardware failures were simulated during this study by using a special card extender containing switches which opened card connector pins to simulate failures. Memory and CPU failures were simulated by modification of the operational software.

The output data from the study defined the time interval between failure introduction and the discrete output command for CAS servo disengagement for the various failure modes simulated. This failure detection time interval was compared to allowable detection time (known detection time which resulted in excessive failure transients) to determine adequacy of the monitoring functions.

Rig Studies -- Studies were conducted to verify the DAFCS dual CPU prototype in-flight monitoring configuration in preparation for flight test in aircraft JA-37-21. These studies utilized the SAAB SCANIA JA-37 "rig", consisting of an aircraft mechanical model, a six-degree-of-freedom digital simulation of the aircraft dynamics, and actual prototype DAFCS electronics.

The purpose of the rig study was to evaluate the safety configuration in a realistic situation to gain confidence before going to the aircraft. These studies investigated most severe failures identified in the prototype safety study and concentrated on sensor and servo monitors. Servo monitoring optimization was also performed using actual JA-37 mechanical system hardware.

Prototype DAFCS hardware was installed in the rig and checked out to ensure that it performed as designed with respect to both hardware and software. The system was then exercised to investigate the risk for nuisance disengagements. Simulated failures were then introduced into the servos and sensors

to evaluate the performance of the applicable monitor and the resulting failure transient. The effect of simulated failures in the outer loops was also evaluated.

JA-37-21 Flight Test -- In addition to functional performance evaluation, JA-37-21 flight test included specific safety tests to --

- Verify nonsusceptibility to nuisance disengagement, and
- Verify that representative in-flight failure transients corresponded to those obtained in simulation studies, thus validating the extensive simulation study results.

Hardware Tolerance Analysis -- DAFCS hardware tolerances were computed from component tolerances for use in monitoring nuisance disengage analysis and for use in establishing preflight test tolerances.

Nuisance Disengage Probability Analysis -- As a basis for this activity, nuisance disengagements are defined as disengagements that occur with all system elements working as designed; i.e., performing within specifications and in the specified environment for the device. When a device such as a rate gyro or servo causes a disengagement as a result of operation outside specified characteristics, this is a failure and not a nuisance disengagement. Within a digital system there are certain elements that do not have performance variations even with specified variations in such things as power supplies, temperature, etc. DAFCS monitoring functions which monitor these elements include the CPU self test, memory sum check, and memory parity error detector. Nuisance disengage probability associated with these monitoring elements should be zero, and this was confirmed by consideration of the design margins of the electronics involved.

Based on these ground rules, the nuisance disengagement analysis considered tolerances in the following devices:

- Dual Sensors
- Servos
- DAFCS Electronics

The analysis identified necessary changes to monitoring thresholds. With changes implemented as needed to satisfy nuisance disengage requirements, quantitative probability of nuisance disengagement associated with each of the stated devices was computed. The resulting DAFCS total nuisance disengagement probability, the sum of the individual parts, was then computed to be within specified limits.

Detailed Level FMEA -- DAFCS functional blocks containing potentially critical failure modes were examined at the piece-part level. Excluding the CPU, whose FMEA is discussed below, more than 3,700 piece-part failure modes were identified, and their probability of failure and failure detectability was investigated.

HDC-301 CPU FMEA -- The HDC-301 CPU contains 15 LSIC's of 10 different types plus more standardized circuitry for the interface functions. Each LSIC has 42 input/output pins. This study examined the effects that failure of the functional logic elements contained in each LSIC would have at the output pins, the impact of these failures on overall CPU operation, if the failure effect would be detected by the CPU self-test program, and how an undetected failure would affect the DAFCS operation.

The CPU FMEA was initiated with analysis of LSIC at the gate level to characterize all gate failures in terms of LSIC output pin "stuck at" faults. A total of 2158 LSIC "stuck at" failure modes were defined to encompass all single gate failures, each involving one, two, three or more LSIC output pins.

All LSIC "stuck at" failure modes involving up to three pins were simulated with actual hardware, and detectability by actual CPU self-test software was verified (see next paragraph). LSIC failure modes affecting four or more pins (666 modes) were analyzed for detectability by CPU self-test software.

HDC-301 CPU Failure Mode Effects Testing (FMET) -- An actual DAFCS electronic unit and a production HDC-301 were used to simulate failures in the CPU to determine the effectiveness of monitor functions in detecting these failures. Failures within the LSIC's were simulated by lifting the leads and introducing open or hardover failures on the output pins. The leads were lifted one, two, and three at a time and combinations of possible failure induced output conditions were evaluated. More than 2,000 LSIC failure modes were simulated in this manner. In addition, another 300 failure modes were evaluated by placing the CPU on an extender such that the connector pins could be opened for the insertion of failure simulations to complete the testing of possible CPU failure modes.

In-flight Monitoring Effectiveness Verification -- Effectiveness of the JA-37 DAFCS single processor preproduction monitoring configuration was verified by simulating the vehicle's reaction to detected and undetected failures. To accomplish this, an actual DAFCS electronics unit was tied into an analog simulation of the vehicle dynamics, the sensors, and the servos; and the flight control software and all monitoring functions were loaded into the DAFCS core memory. System failures were simulated using the extended card and open pin technique and by lifting components on the printed circuit boards, by injection via the analog simulation, and through the use of special software modifications. Aircraft failure transients were observed on the simulation and verified to be within specified limits.

Safety Analysis -- The purpose of the safety study was to analyze all identified functional failure modes of the DAFCS to provide a basis for calculating the probability of catastrophic failure ( $P_{CF}$ ). The analysis identified critical failure modes for each function (i.e., those failures modes that are potentially catastrophic), determined the detectability of these modes by in-flight monitors and ground test, calculated the probability of the mode occurring, and determined the model by which to calculate  $P_{CF}$ .

All identified critical failures fall into one of three classes:

- Detectable in flight and requiring safe disengagement of the DAFCS.
- Defeating the monitor and/or disengage function such that safe disengagement cannot occur, if required.
- Undetectable in flight and potentially catastrophic.

These three classes of failure are represented in the safety calculation model employed in the DAFCS safety analysis.

The safety analysis was accomplished by performing the following tasks, which are shown in flow chart form in Figure 6.

1. Identify functions and determine the safety criticality of each, using a prototype safety study, rig studies, and analysis.
2. Determine the failure modes for all safety critical functions. These were initially identified from the functional level analysis and revised based upon the detailed piece-part analysis.
3. Determine those functional level failure modes that are safety critical; i.e., could cause a catastrophic failure if undetected, could inhibit detection of a critical failure, or prevent disengagement.
4. Determine detectability of all safety critical functional failure modes. This was initially done analytically and subsequently verified by monitoring effectiveness verification and HDC-301 failure modes effects testing. Results were fed back to improve detection by making appropriate changes to the system mechanization, the monitoring mechanization, and/or preflight test methods and procedures.
5. For each critical failure mode, the probability of occurrence was calculated using parts count, failure modes, failure rates, and hazard times. Again, the results were used to determine whether system changes were required to meet the safety requirements.
6. The critical failure probabilities were summed according to the appropriate block of the safety model to which they belonged, and the final value of PCF was calculated.
7. Analysis of CPU failure modes was handled somewhat differently as this analysis was carried down to the individual functional element level within each of the 15 LSIC chips which constitute the CPU. The failure modes of each functional element were analyzed, and an assessment was made as to the criticality and detectability of each mode. Several tests were added to the flight program and preflight test program to detect all identified failure modes.

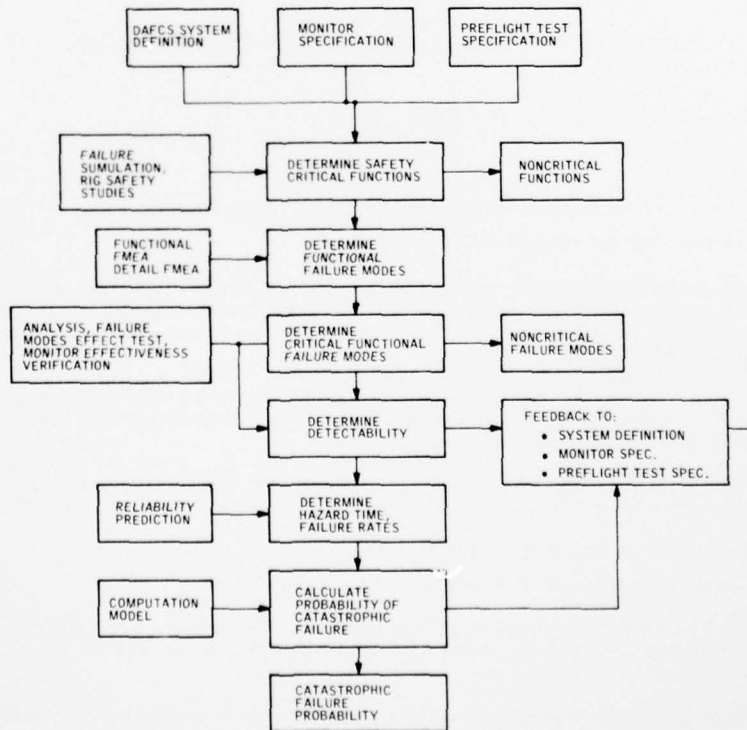


Figure 6. DAFCS Safety Analysis Methodology



Probability of catastrophic failures was thus calculated to be  $0.232 \times 10^{-6}$  for a 1.5-hour mission.

#### Preflight BIT Development

Preflight BIT development was an evolutionary process that resulted in a self-test configuration verified to comply with specified requirements. Important milestones in the preflight test development included these:

- BIT Approved for Flight Test, May 1975 - The decision was made that BIT verification results to date justified reliance on BIT for preflight testing; i.e., comprehensive manual preflight testing could be eliminated in the JA-37-21 flight test program.
- Preflight BIT Approval, 1976 - Verification results approved, redesign actions essentially complete.

Following is a brief description of key activities associated with JA-37 DAFCS preflight BIT development.

Preflight BIT Configuration Definition -- Preflight BIT test methods were devised to facilitate performance test and fault localization to line replaceable unit without the use of external test equipment. Test tolerances were determined such that passage of tests confirmed system performance consistent with aircraft mission requirements. Test methods were documented in a comprehensive DAFCS preflight test method specification.

Preflight BIT Effectiveness Analysis -- Test quality consists of:

- Fault Detection Capability
- Fault Localization Capability
- Performance Verification Capability

A fault is defined as a deviation from specified performance that demands DAFCS corrective maintenance.

Fault Detection Capability -- This is defined as the ability to detect faults by means of available facilities.

$$\text{DAFCS Fault Detection Capability} = 1 - \frac{\lambda_{nd}}{\lambda_{tot}} (\%)$$

where

$\lambda_{nd}$  = failure rate for DAFCS faults not detected

$\lambda_{tot}$  = failure rate for total DAFCS

Fault Localization Capability -- This is defined as the probability of replacing the faulty DAFCS LRU for detectable DAFCS faults.

$$\text{Fault Localization Capability} = 1 - \frac{FFR}{\lambda_d}$$

where

FFR = total DAFCS frequency (failure rate) of faulty replacement

$\lambda_d$  = failure rate for detectable DAFCS faults

The frequency of faulty replacement for the DAFCS is defined as the failure rate for detectable DAFCS faults that results in the replacement of a DAFCS LRU that does not contain the fault. The total DAFCS frequency of faulty replacement is determined from the summation of the frequency of faulty replacement for each DAFCS LRU.

Performance verification capability is established by a test-by-test comparison of estimated operational requirements versus test and measurement tolerances, confirming that test and measurement tolerances are consistent with operational requirements.

Fault detection capability and fault localization capability calculations indicate that both the detection and fault requirements are met for the DAFCS. Calculated values are:

- DAFCS\* fault detection effectiveness: 96.7 percent
- DAFCS fault isolation effectiveness: 99.6 percent

Preflight BIT Verification -- Preflight BIT ability to detect hardware failure was verified by hardware testing. The testing, performed on the SAAB-SCANIA rig using DAFCS prototype hardware, included:

- Analysis of BIT test-measured values to compare with predicted nominals.
- Insertion of faults by opening connector pins, lifting component leads, etc. to exercise each test included in the preflight BIT.

\* Note: DAFCS preflight fault detection effectiveness should not be confused with in-flight potentially catastrophic failure detection effectiveness, which is well above 99 percent.



## CONCLUSIONS

The JA-37 DAFCS -- a single processor Digital Automatic Flight Control System with potentially destructive authority -- has been designed, developed, and verified to have a catastrophic failure probability less than  $1 \times 10^{-6}$  for a 90-minute mission. The verification process was a major element of the DAFCS development program. It involved rigorous analysis, testing, and configuration management. This effort has proved cost effective in that it justified deletion of a redundant second channel in the production DAFCS, significantly reducing production recurring costs.

# DESIGN AND TEST EXPERIENCE WITH A TRIPLY REDUNDANT DIGITAL FLY-BY-WIRE CONTROL SYSTEM

by

Kenneth J. Szalai  
Aerospace Engineer  
NASA Dryden Flight Research Center  
Edwards, California 93523  
USA

Philip G. Felleman  
Division Leader  
Charles Stark Draper Laboratory  
Cambridge, Massachusetts 02142  
USA

Joseph Gera  
Aerospace Engineer  
NASA Langley Research Center  
Hampton, Virginia 23665  
USA

and

Richard D. Glover  
Aerospace Engineer  
NASA Johnson Space Center  
Houston, Texas 77058  
USA

## SUMMARY

A triplex digital fly-by-wire flight control system was developed and then installed in a NASA F-8C aircraft to provide fail-operative, full authority control. Hardware and software redundancy management techniques were designed to detect and identify failures in the system. Control functions typical of those projected for future actively controlled vehicles were implemented. This paper describes the principal design features of the system, the implementation of computer, sensor, and actuator redundancy management, and the ground test results. An automated test program to verify sensor redundancy management software is also described.

## SYMBOLS AND ABBREVIATIONS

A, B, C	channel designations	MVL	midvalue logic
A <sub>s</sub> , B <sub>s</sub> , C <sub>s</sub>	selected signals	N	provisional failure count tolerance
C*	aircraft response parameter, $N_z - \left( \frac{V_{co}}{g} \right) q, g$	N <sub>z</sub>	normal acceleration, positive down, g
CA, CB, CC	hardware comparator designations	Δp	differential pressure, N/m <sup>2</sup>
CAS	command augmentation system	PRI	primary flight control system
CBS	computer bypass system	q	body axis pitch rate, rad/sec
CIP	computer input panel	$\bar{q}$	dynamic pressure, kN/m <sup>2</sup>
DAC	digital-to-analog converter	RAV	remotely augmented vehicle
DFBW	digital fly-by-wire	RM	redundancy management
g	acceleration of gravity, m/sec <sup>2</sup>	s	Laplace transform variable
IFU	interface unit	SAS	stability augmentation system
K <sub>C*</sub>	pitch CAS loop gain, deg/sec/g	V <sub>co</sub>	crossover velocity, m/sec
K <sub>XF</sub>	flap-to-stabilizer crossfeed gain, deg/deg	VTAS	true airspeed, knots
K <sub>1</sub> , K <sub>2</sub> , K <sub>3</sub> , K <sub>4</sub> , K <sub>5</sub>	control system gain designations	α	angle of attack, deg
LVDT	linear variable differential transformer	α <sub>L</sub>	limit angle of attack, deg
		ω <sub>n</sub>	natural frequency, rad/sec
		ζ	damping ratio, percent of critical

## 1.0 INTRODUCTION

The National Aeronautics and Space Administration (NASA) is conducting research in digital fly-by-wire (DFBW) flight control for aircraft. The primary impetus for this work has come from the projected flight control system requirements for advanced military and commercial aircraft, particularly in the area of active controls. Such systems must have extremely high levels of reliability and computational capacity.

From 1972 to 1973, the NASA Dryden Flight Research Center (DFRC) conducted flight tests of a DFBW control system in an F-8C aircraft. The F-8 DFBW aircraft was equipped with hardware developed for the Apollo

lunar module guidance, navigation, and control system, and, from its first flight, was flown with the basic mechanical control system removed. A triplex analog fly-by-wire control system provided emergency backup control. This program successfully demonstrated the feasibility of DFBW control for aircraft (refs. 1 to 4).

Since that program, analog fly by wire for production aircraft has become a reality in the F-16 airplane (ref. 5). In addition, digital flight control has been actively investigated by government and industry in the United States (refs. 6 to 17) and in Europe (ref. 18). The space shuttle orbiter will also depend on a redundant digital fly-by-wire system for primary flight control (ref. 19). However, a fault-tolerant, flight-critical DFBW system has yet to be flight tested. NASA is currently conducting research in several technology areas related to advanced DFBW control. This research program is planned to culminate in the flight test of a fail-operative, fail-safe triplex DFBW control system in the F-8C aircraft. The primary objective of the program is to provide a design base for future practical DFBW control systems. The specific tasks related to the flight research program are to formulate and evaluate in flight (a) hardware and software redundancy management concepts for sensor, computer, and actuator systems; and (b) digital flight control laws of the type projected for advanced aerospace vehicles. In addition to these tasks, selected redundancy management concepts for the space shuttle orbiter will be flight tested on the F-8C aircraft. To do this, software routines developed for the orbiter's primary and backup flight control systems will be executed in the F-8 DFBW system.

Flight hardware has been installed in the F-8C test bed, and final software verification and systems integration testing has been completed. This paper describes the F-8 DFBW system design and mechanization, and summarizes the ground test experience with the system.

## 2.0 SYSTEM DESCRIPTION

### 2.1 F-8C Testbed Aircraft

The F-8C aircraft (fig. 1) is a single-engine, single-place U.S. Navy fighter capable of supersonic flight. The aircraft has a two-position variable incidence wing for reducing fuselage attitude during the landing approach.

The modifications to the F-8C aircraft, which were entirely internal, consisted of the addition of hardware and software subsystems. The mechanical linkages connecting pilot controls with surface actuation systems were removed in the first phase of the program.

### 2.2 Overall System Mechanization

The overall mechanization of the F-8 DFBW control system is shown in figure 2. A triplex digital computer set containing the control law and system redundancy management software communicates with a specially designed interface unit (IFU). The IFU processes input data, which consist of pilot commands and aircraft sensor signals, and output data, which consist of surface commands, cockpit displays, and telemetry data. Surface commands are routed through a switching mechanism to the servodrive electronics and then to the force-summed secondary actuators, which are installed in series with the existing F-8C power actuators. There are five actuator sets: one for each aileron and horizontal stabilizer surface and one for the rudder.

The triplex analog computer bypass system (CBS) provides the pilot with an emergency unaugmented command path to the control surfaces in the event of a total primary digital system failure. This path was provided primarily to protect against a common-mode software failure in the infant stages of flight test. The switching mechanism allows either the primary system or the bypass system to drive the secondary actuators based on pilot selection or fault status.

Electrical power is provided to three independent flight control buses by an engine-driven dc generator. Each bus is protected by a 40-ampere-hour battery, which would allow approximately 90 minutes of operation in the event of a loss of generator power. Secondary actuator hydraulic power is provided by the aircraft's three hydraulic systems, each of which supplies one of the triple chambers of each actuator.

### 2.3 Hardware Subsystem Layout

The major hardware elements of the DFBW system and their locations in the F-8C aircraft are shown in figure 3. The triplex digital computers and IFU are on a removable pallet assembly. An encoder/decoder, which is used to transmit cockpit display information to and from the IFU, is in the nose of the aircraft. In addition to the conventional center stick and rudder pedal controls, a two-axis, limited-displacement sidestick is available to the pilot.

The computer bypass system and servodrive electronics are contained in three interchangeable boxes. The sensor pallet contains triply redundant rate gyros and accelerometers. Other sensors, including attitude gyros and angle of attack and sideslip vanes, are at separate positions in the aircraft.

The secondary actuators are as near as possible to the F-8C power actuators. The shaft of each secondary actuator is linked through mechanical gearing to the metering valve of its associated power actuator at the point where the pilot's controls would normally be connected.

Four cockpit panels provide the pilot with primary system controls and displays (fig. 4). The mode and gain panel gives the pilot pushbutton control over the primary and bypass modes on an individual axis basis. Three five-position gain switches can be tied, through software, to any parameter that requires pilot evaluation. Selection of pilot relief modes is made through the digital autopilot panel, which utilizes magnetically held switches. Caution and warning messages are displayed on the annunciator panel. Four of the 20 annunciation lights have built-in switches to perform software reset functions. The computer input panel provides the pilot with a way to initiate preprogrammed software sequences, such as the preflight test program. This panel is also intended for in-flight use in selecting certain control system options. Two thumbwheel switches are used to select the program. A three-digit display indicates the selected program. When the enter switch is depressed, the displayed program is executed.

## 2.4 Primary Digital Flight Control System Mechanization

A functional block diagram of the primary digital flight control system is shown in figure 5. The three channels (A, B, C) are identical. A variety of sensors and discretes is used by the primary system. Table 1 lists each input sensor, its redundancy level, and its signal type. Analog sensors are converted through a 12-bit analog-to-digital converter. The only digital input is from the altimeter. Input discretes and their redundancy levels are listed in table 2.

Each channel conditions and converts sensor data associated with that channel. The data are placed in the buffer memory of that channel and in the buffer memories of the other two channels by way of the serial data buses. Thus, each computer contains an identical set of redundant sensor data. Sensor redundancy management is performed in software and the selected sensor signals are used in the control laws to compute surface command signals. The serial data buses are also used by the computers to transmit and receive status information for computer redundancy management. The intercomputer discretes are used only to synchronize the computers.

The IFU output consists of surface position commands to the horizontal stabilizer, ailerons, flaps (symmetrical ailerons), and rudder; discretes to the cockpit panels; and telemetry data. The panel discretes are transmitted serially to the encoder/decoder for distribution within the cockpit. The serial telemetry data, which consist of internal digital computer data, are transmitted to an onboard tape recorder. Surface commands are generated through a 12-bit digital-to-analog converter to an electronic switch. When the primary system is active these signals are passed directly through the switch to hardware midvalue logic (MVL) select circuits. There is a set of three MVL modules for each of the five secondary actuators. The function of the comparators is described in the section on actuator redundancy management.

The secondary actuators use high-gain two-stage servovalves to control hydraulic pressure of  $2.07 \times 10^7 \text{ N/m}^2$  ( $3000 \text{ lb/in}^2$ ) across each of three pistons. Force summing occurs along the common output shaft, which is mechanically linked to the metering valve of the existing dual-tandem F-8C power actuators. The nominal characteristics of the secondary and power actuators are given in table 3.

## 2.5 Digital Control Computer

The computers used in the digital flight control system are general purpose, stored-program machines that use a microprogrammed instruction set. In addition to the two sets of eight fixed-point general registers, the computers contain eight hardware registers for floating-point operations. The major computer characteristics are listed in table 4. The F-8 DFBW configuration, which will be flown initially with 24,576 words of memory, allows expansion to 32,768 36-bit words, including store protect and parity bits. The computers are cooled by individual fans.

## 2.6 Interface Unit

The IFU consists of all of the equipment necessary to process and condition the input and output signals of the three digital flight computers. The major functional design requirements were to provide a simple interface with the programmer for most input-output functions, to provide interchannel isolation for prevention of common-mode failures, to allow expansion to a fourth channel, and, finally, to provide a high degree of flexibility in the research application. To meet the F-8C installation envelope requirements, the three IFU channels were packaged within a single enclosure, but each channel had complete electrical isolation. The IFU is composed of individual modules with unique functions.

### 2.6.1 Control

Each channel of the IFU contains a microprogrammed controller that decodes computer commands, differentiating the various types of input and output requests; controls data transfers between the various fields of IFU; and performs validity checking of control and data signals.

The control program is contained in a programed read-only memory in each IFU channel. The microprogram flow is shown in figure 6. The microprogram cycles in a wait state, searching for a command from the computer to begin a transaction. Upon receipt of the appropriate signal, the first word transmitted is decoded to distinguish the type of transaction, and appropriate flags are set in the interface unit to alert or initiate activity. A dialogue of signals and data between the controller and the computer continues until the transaction is complete. A timeout function in the control section guards against response delays. If a microprogram step waits for a response for longer than 32 microseconds, the transaction is terminated and an input-output error alarm is issued to the computer.

### 2.6.2 Computer Interface

Each IFU channel communicates with its associated computer by way of a dedicated interface, which consists of parallel input and output buses for data and a number of dedicated function interfaces that control the input-output section of the computer and the signals to the interface unit.

All computer-to-IFU data transfers are computer initiated and are accomplished by way of a computer mode that minimizes the requirement for the computer software to actively participate in the input-output process. During this data transfer time, other operations can take place in the computer, but the input-output section is not available for those operations.

### 2.6.3 Encoder/Decoder

Functionally, the encoder/decoder can be considered a part of the IFU although it is physically separate. One function of the encoder/decoder is to format switch inputs from the cockpit panels into several words and to transmit these to the IFU by way of a serial digital data bus. The other function of the encoder/decoder is to receive words transmitted from the IFU, process them, and drive the cockpit displays. As with the IFU, the encoder/decoder is arranged in a channel configuration to provide fault tolerance.



AD-A041 042

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT--ETC F/G 1/3  
INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS.(U)  
1977

UNCLASSIFIED

AGARD-06GRAPH-224

NL

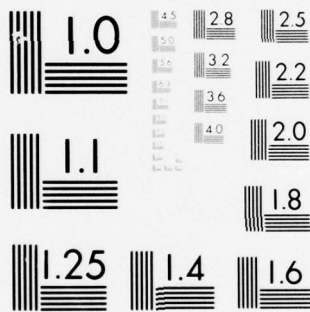
4 OF 4

AD  
A041042



END

DATE  
FILMED  
7-77



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

#### 2.6.4 Data Input

Each channel is capable of processing the input data listed in tables 1 and 2 and consists of 32 analog signals and 10 discrete words of 16 bits each. Five of these discrete words are serial data. All of the analog inputs are prefiltered in the IFU.

#### 2.6.5 Data Output

Each channel can transmit 10 12-bit analog words and five discrete words of 16 bits each. Three of the discrete words are sent in serial form. In the flight configuration, only four of the analog outputs are assigned: commands to the horizontal stabilizer, the ailerons, the flaps (symmetrical ailerons), and the rudder.

#### 2.6.6 Buffer Memory

Each channel of the IFU uses three buffer memories to accumulate data from the resident channel and from the other two channels. The memories are used to store converted sensor data and computer-to-computer-transferred data (crosslink). Each of the memories, which are of the first-in-first-out type, has a capacity of 64 16-bit words. A buffer memory is loaded by control pulses from the sending channel and unloaded by control pulses from the receiving channel.

#### 2.6.7 Interchannel Data Transfer Control

So that each channel has access to all of the data, any data received by one channel are simultaneously transferred to the buffer memories of the other two channels, as shown in figure 7. When a channel is unpowered, it cannot participate in any interchannel transactions; therefore, each operational computer commands its IFU to ignore any unpowered channel.

#### 2.6.8 Data Recording

Each channel has a separate buffer memory for storing user-selected digital data for onboard recording. The system is capable of recording 1000 32-bit data words per second per channel.

#### 2.6.9 Failure Detection

Each channel contains failure detection circuitry to monitor the IFU and computer fault status. Figure 8 shows the fault detection functions that result in a channel fail declaration. The channel fail signal is set by any of the following conditions:

- (a) Failure of the IFU clock
- (b) Out-of-tolerance condition on the +28,  $\pm 15$ , or  $\pm 5$  volts dc power
- (c) Absence of computer input-output activity for longer than 57 milliseconds
- (d) Computer-generated fail discrete caused by the built-in test equipment of the computer or by software fault detection
- (e) The agreement by two operational channels that the third channel has hard failed

Each channel forms the hard-fail declaration signals for the adjacent channels from software-generated discretely. These discretely are interlocked with the hardware fail signal to prevent a failed channel from declaring a hard fail in another channel.

#### 2.7 Software Structure

The F-8 DFBW software is built around a flexible executive program that can schedule jobs on the basis of a clock or event interrupt and priority. The flight control program is written entirely in assembly language. Fixed-point arithmetic is used in the sensor redundancy management routines to minimize execution time; floating-point arithmetic is used exclusively in control law routines. The sequence of the F-8 DFBW software is shown in figure 9, along with measured execution times for one minor cycle.

An executive interrupt initiates the minor cycle. After the computer synchronization and computer redundancy management jobs are completed, the next minor cycle interrupt is scheduled. The input data are then read and used in the redundancy management and control law routines. These two routines are each subdivided. To minimize transport delays, only those computations necessary to determine surface position commands are computed prior to the output job. The data recording processor collects and transmits 20 full words of data during each minor cycle. The computer input panel (CIP) processor handles pilot requests for initiation of preprogrammed functions, which include control law options, display of fault status, and initiation of the preflight test program. The computer self-test is the lowest priority job and includes arithmetic unit checks, instruction tests, and a memory sum check.

In addition to the software for the active flight control program, other software modules exist for ground test use only. These include the preflight test program, special real-time display programs for use with a ground-based cathode ray tube, and a crosslink loader that enables the ground crew to load two computers from the third. The software memory allocation is given in table 5.

### 3.0 SYSTEM REDUNDANCY MANAGEMENT

The responsibility for redundancy management of the primary system is shared by the hardware and software functions. This section describes the redundancy management designs for the computers, sensors,

discretes, and actuators. The system design ground rules require fail-operative, fail-safe capability for critical functions and fail-safe capability for noncritical functions.

### 3.1 Computer Redundancy Management

Several features were implemented within the primary system to detect hardware or software failures within a computer or its interface and to isolate the failed system so that the remaining processors could continue normal, transient-free operation. In addition, each processing system was designed to be tolerant of power transients, which could cause momentary cessations of computation. The main design features involved in computer redundancy management are synchronization and error handling.

#### 3.1.1 Synchronization

To insure that each processor handle input and crosslink data at the same two points in the computation cycle, the computers are software synchronized to begin each 20-millisecond cycle within 10 to 50 microseconds of each other. No other synchronization points exist within the computation cycle. Output commands are sent independently by each channel when they are available. Because of the close tolerance on the synchronization point at the beginning of each cycle and the particular minor cycle structure used, output synchronization is unnecessary. The close synchronization also provides an excellent measure of channel health and hence becomes an important element in computer fault detection.

To synchronize the three computers, discrete signals are sent from each computer, through the IFU, to each of the other two computers (fig. 10). At the beginning of each cycle, an internal computer clock interrupt occurs and the computer issues a discrete signal to each of the other computers. The computer then reads the discretes it has received from the other computers. If discretes are present from both computers, the discrete is reset, and a second read is performed to insure that the other computers have reset their discretes. The computer clock is then reset to interrupt at the next cycle time. If, after a short wait to allow for skew between processors, one computer fails to synchronize with the other two, the two remaining computers exit the sync program and continue normal processing.

One subroutine of the synchronization program is the intercomputer transfer, or crosslink, of a small, selected set of data to verify that each computer agrees with the state of each of the others. When each computer reaches the run state, six 16-bit words are exchanged with the other computers. If a computer finds a disagreement, it requests a restart. The synchronization and crosslink algorithms are designed to operate when any two of the three or all three computers are powered. Dual discretes are used to identify faults in the discrete signals. If the two discretes do not agree, a synchronization failure is declared.

#### 3.1.2 Error Handling

Five types of computer/IFU errors are recognized by the system, and each error type results in a different action. The error types are alarms, restart requests, soft failures, hard failures, and self failures.

##### 3.1.2.1 Alarms

Alarms are generated by either hardware- or software-detected errors. Some of the typical alarms are as follows: illegal instruction, store protect error (that is, an attempt to use data as instruction or vice versa), illegal address, parity error, computer self-test error, crosslink error, and IFU error. When an alarm indicates a serious and possibly permanent fault, a restart is requested in an attempt to clear the fault. Other alarms are merely logged for postoperation analysis.

##### 3.1.2.2 Restart requests

A restart is an online reinitialization of the software to bring all computers into agreement as to the state of the system. Some of the typical causes of restarts are power transients, synchronization failures, certain alarms, crosslink failures, and disagreement among real-time counters.

Whenever a restart is requested, the three computers, by way of the crosslink, exchange enough data to guarantee that they are in agreement. This transmission includes the choice of the computer considered to have valid data and the data to be used by all computers. The exchanged data are 58 16-bit words, and include such items as sensor failure history, control law parameters, and pointers to align major cycle sequences.

To prevent continuous restart requests caused by a failure in the system, each computer maintains a count of all restart requests. If the number of restart requests made by any computer exceeds a prescribed tolerance, that computer is declared hard failed and its requests are henceforth ignored. The entire restart process takes approximately 4 milliseconds from recognition of request to resynchronization.

##### 3.1.2.3 Soft failures

A soft failure is a reversible declaration by any computer that one (or both) of the other computers appears to be unpowered. A soft-failure declaration results when neither synchronization nor crosslink is received from the offending channel for 10 consecutive passes. After declaring a channel soft failed, the remaining channels bypass the search for synchronization and data from the failed channel. The failure declaration is cleared when crosslink data from the soft-failed channel reappear.

##### 3.1.2.4 Hard failures

A hard failure is an irreversible declaration by any computer that one (or both) of the other computers is powered but not functioning. Two conditions can lead to a hard-failure declaration: the absence of crosslink data from a computer that is correctly synchronized for 10 consecutive minor cycles, and an excessive number of restart requests in a given time period. If two computers agree that the third is hard failed, a channel failure is declared. The actuator redundancy management then reconfigures to prohibit the failed computer from providing control outputs.



### 3.1.2.5 Self failures

A self failure is an irreversible declaration by a computer that its own channel is no longer functional. A self failure may be declared when a channel has too many input-output errors, is declared failed by the other two computers for 10 consecutive minor cycles, finds the other two computers hard failed, or has an excessive rate of restart requests. After a self-failure declaration, all active computing in that computer is terminated and the computer goes to an idle state.

## 3.2 Sensor Redundancy Management

Sensor redundancy management (RM) is divided into two phases (RM-1 and RM-2) that are executed at different times in the computational cycle. The only function of RM-1 is to obtain the best estimate of the actual parameter value based on the available multiple sensor inputs. RM-1 is executed during every minor cycle between the completion of the data input read job and the control law computations. RM-2, which is designed for fault detection and identification, controls reconfiguration of the RM-1 select logic. RM-2 is executed late in the minor cycle and may be performed at a slower rate than RM-1 in cases where isolation of a faulty sensor is less time critical.

A typical triplex RM algorithm is shown in figure 11. RM-1 begins as a midvalue select mode, changes to an averaging algorithm after the first hard failure, and finally degrades to a default output value after the second failure. A hard sensor fault is declared by RM-2 when a sensor differs from the selected value by an amount greater than the allowable tolerance for a given number (N) of consecutive passes. Failure-status logic monitors the results of the tracking test and through hard-fail flags causes the mode or function using that sensor to be inhibited. For example, should the entire roll rate gyro set be lost, roll stability augmentation system (SAS) would be inhibited. In some cases annunciation is given to the pilot when an entire sensor set has been lost. The first failures of sensors in a triplex set are not annunciated to the pilot.

## 3.3 Discrete Redundancy Management

A simplified representation of the discrete RM algorithm is shown in figure 12. The mask bits are initially all set. During RM-1, the three input discretes are AND'ed with their respective mask bits and the event output is determined by a majority vote. During RM-2, the event output is compared with the masked discretes and disagreements are declared to be provisional failures. When a specified number of consecutive disagreements has occurred, the failure detector clears the corresponding mask bit and declares that input hard failed. Because the majority voter, on subsequent RM-1 passes, always sees a zero for that input, the output in effect becomes the AND of the remaining two inputs. If the two remaining inputs disagree for N passes, the output is set to zero and the failure status logic sets the event failure flag.

A novel approach has been taken in the software implementation of this algorithm. Rather than processing each trio of discretes separately, up to 32 sets may be processed simultaneously by operating on packed full words with logical instructions. The A inputs are contained in one word, the B inputs in a second, and the C inputs in a third. The three masks are also 32-bit full words, one assigned to each input word. Similarly, the failure flags, disagreement flags, and output are all 32-bit full words. In each case, a given bit position in the full word is dedicated to a specific event. Great economy in execution time has been achieved with this technique.

## 3.4 Actuator Redundancy Management

Computer output and servoactuator RM is handled entirely in hardware. As shown in figure 5, the command path to the actuator from either the primary or bypass control system is controlled by both pilot command and failure status signals. Table 6 illustrates the switch logic for sample events that could occur while the primary system is active.

If comparator CA indicates a discrepancy between the selected midvalue and that channel's input command, the A switch transfers to the CBS state. In this mode, the A input is forced to track the MVL output. If a second comparator, CB, indicates a failure, both the B and C switches transfer to the CBS state, placing the bypass system in complete control of that secondary actuator. No single failure can cause all three channels to transfer to the bypass mode.

Fault detection logic mechanized in the primary digital subsystem in both hardware and software can also cause the switch to change state. A single digital system failure in channel C, for example, causes switch C to transfer to the CBS state. A second failure, such as channel B, causes both A and B switches to transfer to the CBS state. In this case, all three axes of control are transferred to the bypass system.

Note from figure 5 that for an unfailed condition or for channel element failures prior to the MVL module, the three actuator commands generated by the primary or bypass systems are identical. If an MVL module, a servo-electronics element, or a secondary actuator element fails, fault detection occurs within the servoelectronics module.

Figure 13 is a functional block diagram of a single channel of one of the secondary actuators and the electronics of that channel. The servoamplifier signal is the sum of the position command, the shaft position feedback, and a nonlinear function of differential pressure ( $\Delta p$ ). The latter signal ( $\Delta p$ ) is designed to minimize the low-frequency force fight that occurs in the high pressure gain system. The circuit acts to drive the  $\Delta p$  of each valve toward the midvalue of all three. The bandwidth and authority of the  $\Delta p$  feedback are limited to allow detection of faults that could be hazardous. If the  $\Delta p$  of one valve differs from the midvalue-selected  $\Delta p$  by approximately  $8.3 \times 10^6 \text{ N/m}^2$  (1200 lb/in<sup>2</sup>) for 400 milliseconds, a fault is declared which disables the engage solenoid, dumps supply pressure to return, and opens a bypass path around the piston in the failed channel. Faults are annunciated in the cockpit and reset capability is provided to the pilot. A second failure in the same actuator results in that actuator's being turned off. Mechanical centering springs move the disabled actuator to a safe static position.

#### 4.0 CONTROL LAWS

The control laws mechanized for the F-8 DFBW aircraft were selected to include several functions projected for use in future active control applications that would require full-time, full authority control (ref. 20). The pitch, roll, and yaw axes have multiple pilot-selectable modes. Each axis contains a DIRECT mode which provides unaugmented control of the aircraft. In this mode, the surface command is the sum of stick and trim components. A pitch SAS mode uses scheduled pitch rate feedback to improve short period damping. These modes provide less complex configurations in the event of failures in certain sensors in the more highly augmented pitch command augmentation system (CAS) and lateral-directional SAS modes which contain the active control functions. Only the pitch CAS and lateral-directional SAS modes are described in detail here.

All inner loop control law functions are computed at the minor cycle rate of 50 samples per second. Autopilot functions, scheduled gain updating, and other less time-critical operations are computed at the major cycle rate of 12.5 samples per second. Both the minor and major cycle rates can be altered by single-entry changes in the executive timing routine. Filter coefficients, however, are not automatically changed with changes in sample rate.

The control laws were designed in the continuous domain. The Tustin transform was used to obtain discrete versions of the continuous filters. At a sample rate of 50 samples per second, the Tustin transform and the matched bilinear transform described in reference 21 yield nearly identical results. The control law modules are functionally independent of the redundancy management software and use only midvalue-selected sensors and majority-voted discretes.

##### 4.1 Pitch Command Augmentation System Mode

The pitch CAS mode includes several typical active control functions. If a new airplane were being designed, such control laws would be considered in the initial trade studies leading to the ultimate configuration. No structural or aerodynamic changes were made to the F-8C aircraft in conjunction with these control laws; therefore, it is not possible to extract performance trade-off information directly from the F-8 DFBW flight test results. The control law research objective of the F-8 DFBW program is to evaluate the mutual interactions of the control functions in a full authority, flight-critical digital implementation.

Figure 14 illustrates the command augmentation functions implemented in the CAS mode. The basic controller combines prefiltered stick deflection with pitch rate and normal acceleration. The resulting signal is routed to the actuation system by way of a variable gain ( $K_{C*}$ ), which is scheduled with the dynamic pressure derived from altitude and Mach number. To minimize excessive stick forces during large changes in airspeed, neutral speed stability is provided by an effective forward-loop integration. The integration is mechanized by the cancellation of the position feedback signal of the secondary actuators at low frequencies.

A significant feature of this control law is that it was designed through the application of linear optimal control theory at selected flight conditions. Specifically, the motion variable  $C^*$ , which is defined as  $N_z - \left(\frac{V_{co}}{g}\right) q$ , was compared with the output of a linear second order command model with a natural frequency of 7.4 radians per second and a damping ratio of 0.91. Minimization of a quadratic cost functional, which consists of the weighted sum of the  $C^*$  error, its integral, the elevator rate, and the elevator command, resulted in a control law that is a linear combination of the assumed state variables. The latter control law was examined for low-gain loop closures and possible pole-zero cancellations. The result of these simplifications is the pitch CAS control system shown in figure 14.

The design specifications for the longitudinal control system required that angle of attack be limited. The implementation of the angle-of-attack limiter and its integration with the basic  $C^*$  control law are shown in figure 15. The design of this control law was also accomplished by the use of optimal control theory. The quadratic cost functional included pitch rate, angle of attack and its integral, elevator rate, and elevator command. As in the case of the basic  $C^*$  design, the control law obtained from the optimal control theory at selected flight conditions was rearranged to yield the final configuration. The resulting control law uses angle of attack and high-passed pitch rate. The angle of attack is referenced to the value of  $\alpha_L$ , the limit angle of attack.

As shown in figure 14, the switching logic selects the more positive (nosedown) command of either the normal CAS command or the  $\alpha$ -limiter command. The high-passed pitch rate provides an anticipatory term when the limit angle of attack is approached rapidly.

Direct lift produced by symmetrical aileron deflections is utilized for drag reduction in maneuvering flight and for ride smoothing in turbulence. The complementary structure of the direct lift mode is illustrated in figure 16. Low-frequency symmetrical aileron deflections are commanded by pitch rate, which is routed through a first-order lag with a time constant of approximately 2 seconds. This mode provides a trailing-edge flap deflection schedule that was chosen for a maximum lift-to-drag ratio. Ride smoothing is accomplished by relaying rigid-body normal acceleration to the symmetrical ailerons through the gain  $K_I$  and a high-pass filter with a time constant of 0.4 second.  $K_I$  is scheduled with dynamic pressure. The high-pass filter attenuates the command during steady maneuvers. Changes in pitching moment due to symmetrical aileron deflection are canceled by use of a crossfeed signal through  $K_{XF}$ , which is chosen to be the ratio of the pitching moments produced by unit deflections of the horizontal tail to those of the symmetrical ailerons. Reductions of 10 to 30 percent in rigid-body normal accelerations due to gust disturbances are predicted (ref. 20).

##### 4.2 Lateral-Directional Stability Augmentation System Modes

Figure 17 illustrates the mechanization of the augmented modes for the lateral-directional SAS modes. Although the roll and yaw SAS modes are individually selectable by the pilot, they are treated collectively in this discussion. The criteria for the design of this system included improved damping of the Dutch roll oscillatory mode and good directional stability and turn coordination at all usable angles of attack. Application of the

linear quadratic optimal control algorithm at selected flight conditions yielded a feedback gain matrix with non-zero gain on every state variable to every control input. A separate algorithm, described in reference 20, was used to drive to zero the gains for which implementation was impractical. In the resulting control mode, high-passed yaw rate provides increased Dutch roll damping with no steady-state effect. Turn coordination is provided by compensated lateral acceleration and an aileron-to-rudder interconnect. Minimization of steady-state sideslip is achieved by feeding the integral of lateral acceleration to the rudder. The gains are scheduled with angle of attack to yield good performance in all maneuvering conditions.

#### 4.3 Autopilot Functions

In addition to the inner loop functions, the control law set includes several autopilot functions—attitude hold, Mach hold, altitude hold, and turn command. Control stick steering is available to allow pilot control through the autopilot.

#### 4.4 Remotely Augmented Vehicle Mode

In addition to the control modes described above, a special remotely augmented vehicle (RAV) mode has been mechanized to provide increased research capability (fig. 18). Pilot stick commands and vehicle motion sensors are telemetered to a ground station as 10-bit words. Highly speculative and advanced control concepts are to be programmed in FORTRAN on the ground-based minicomputer. The control surface commands are then transmitted to the F-8C aircraft, where an uplink receiver and decoder preprocesses the commands and sends them to the primary digital system. Pilot-selectable RAV modes in the digital computer software test the validity of the surface commands and route them to the actuator drive electronics.

The RAV mode is an asynchronous job which is executed at the rate of 100 samples per second. Control of the aircraft through the RAV mode is selectable by axis. In the event of an uplink fault or an invalid command, the control system automatically reverts to the SAS modes.

The RAV mode of operation will be used to evaluate concepts such as advanced adaptive control, where the researcher will be able to modify the control laws easily without the possibility of adversely affecting the basic primary control law software.

### 5.0 GROUND TEST EXPERIENCE

The majority of the system integration and software verification testing was accomplished in the NASA DFRC iron bird facility (fig. 19). A modified F-8 airframe provides a complete test bed for the DFBW system. The systems in the iron bird include a primary digital system pallet, an encoder/decoder, a computer bypass system, actuator drive electronics, a complete set of secondary and power actuators, three independent hydraulic systems, three independent electrical systems, and complete cockpit panels. The iron bird is interfaced with an all-digital nonlinear representation of the F-8C aerodynamics, which allows complete pilot-in-the-loop simulation.

To facilitate ground testing, several real-time displays were developed for use with a ground-based cathode ray tube and printer to display either preprogrammed data lists or user-specified data lists. Test data are primarily collected by the data recording channel. The nine-track tapes generated at the iron bird are processed directly on the NASA DFRC central computer.

More than 1500 hours of ground tests have been accumulated on the primary digital flight control system, using both a breadboard triplex system and the actual flight hardware. This testing has included both software verification and system performance evaluation.

#### 5.1 Sensor RM Verification Experience

Considerable experience has been gained from the software verification testing of the F-8 DFBW sensor RM algorithms. A philosophy of exhaustive testing was adopted at the outset because of the varied types of sensors involved. Complete testing of the RM algorithm for all sensors proved to be a sizeable effort and spurred the development of a software tool to speed the verification process and reduce the engineering test time required.

##### 5.1.1 Test Requirements

For each analog sensor, several characteristics of the redundancy management algorithm had to be tested. First, three measurements had to be made: the tolerance level at which provisional failure counters start counting, the value N at which a hard failure is declared, and the default value of the output following the second failure. Second, proper operation of both the midvalue-select and averaging modes had to be verified. Third, proper processing of the RM-2 counters and flags had to be checked. Fourth, control law inhibits and annunciation to the pilot had to be demonstrated to complete the test.

##### 5.1.2 Test Method

To obtain the above test results, a means was required to excite the RM algorithms with simulated sensor inputs. This excitation was provided by the RM support software package, which was developed at DFRC specifically for the F-8 DFBW project. The software package is written in FORTRAN and is executed with the real-time simulation program used to support the iron bird testing. Using the software package, any of the wave forms shown in figure 20 can be injected independently into the inputs of the RM algorithm. The package is controlled from a remote terminal as shown in figure 21, which permits real-time selection of the wave form, controlling parameters, and engage and bypass status.

Any sensor can be selected by way of the terminal to be modeled as a redundant sensor. The particular wave form and the characteristic parameters, such as drift rate or bias, are selected for each channel. The contaminated channels can then be individually engaged or bypassed through the use of the terminal.



To efficiently test all elements of the RM algorithm, a composite test profile was programed into the RM support software program to exercise all of the RM functions. Within a span of 33 seconds, six tests are performed. These tests are controlled by user-inserted tolerance levels and an N value. After the RM support software program is engaged, the test begins with impulse bursts above the tolerance level, but within a pulse width of N-1 minor cycles so as not to trigger hard failures. The second segment introduces transients below the tolerance level to verify that no provisional failures are declared. The third segment forces the midvalue select logic to periodically switch among the three inputs. The fourth and fifth segments force a hard failure of sensor C followed by the hard failure of a second sensor, which is expected to fail the entire sensor set. The sixth test verifies the correct default value for arbitrary sensor inputs.

This sequence has also been used to test dual sensors. Two additional automated profiles have been incorporated into the RM support software program. One of the automated profiles produces three-channel synchronous sawtooth wave forms to test those sensors which have rate reasonability tests at the output of RM-1. The second automated profile was designed to test dual and triplex discretes with fault impulses that increase in width by one minor cycle each iteration.

### 5.1.3 Test Results

Figure 22 is an example of the digital plots obtained using the RM support software program to simulate a typical triplex analog sensor. The sensor inputs and selected output have been normalized to the tolerance level and, for the case shown, the provisional failure count N is five. Test segment 1 exercises the algorithm with repetitive impulse bursts 50 percent above the tolerance level to verify that the fault detection logic is operating properly. The output remains unaffected while the provisional fail counters tally down to one, stopping short of the failure declaration level of zero, and then reset to five where the input returns to normal.

The second test segment verifies that no provisional failures are tallied for sensor excursions within the tolerance limits. The output reflects the passage of the 1-hertz sinusoids when they appear on two or more of the inputs.

The third segment tests proper operation of the midvalue select module. The output wave form is clipped as sensors A and C alternately become the midvalue. Once again, no activity is seen in the failure counters.

The fourth test segment forces a failure in sensor C by applying ramp inputs to channels A and B to a magnitude 1.5 times the tolerance level, thus simulating a null failure in the C sensor. When the ramp inputs reach the tolerance level, the channel C counter begins to tally down. When the counter reaches zero, the C sensor is declared failed, the algorithm switches from the midvalue to the average of the A and B sensors, and the C counter is latched at -1 to signify that reconfiguration has been completed.

The fifth test segment introduces a divergence between the A and B sensor inputs to cause a second failure. When the difference between the output and either of the inputs exceeds the tolerance level, the failure counters begin to tally down. Because sensor input A is tested first by the algorithm, its counter reaches zero first. When this occurs, the output is clamped to the default value (in this case, zero), the A counter is set to -1, and all RM activity ceases for this sensor. Test segment 6 verifies that activity has ceased for sensor A and applies a large signal on all three sensor inputs to assure that subsequent sensor A activity is not passed by the algorithm.

The RM support software program is also used to conduct failure mode and effects demonstrations during piloted simulation and to evaluate overall performance of the sensor RM algorithms in various simulated environments.

## 5.2 Computer RM Test Experience

The synchronization, restart, and fault detection performance of the primary system has been extensively evaluated during the ground test phase of the program. The performance of the system has been substantially improved by the changes made during the ground tests.

### 5.2.1 Synchronization Performance

The specification for maximum synchronization skew was 200 microseconds. In practice, much better synchronization has been achieved. Figure 23(a) shows the normal sync discrete pattern observed on an oscilloscope. The leading edge of the pulse indicates the setting of the intercomputer discretes by each channel. The trailing edge of the pulse represents the time at which each computer has verified the correct sync state of the other two computers. The computers search for the run state of the other computers before leaving the sync job. Total execution time of the sync job is approximately 230 microseconds. The average skew time between computer discretes at the end of the sync state is 10 microseconds. Occasionally, because of a variation in the computation time, a computer arrives too late to be detected by the other two computers on the first check. In this case, as shown in figure 23(b), an additional pass through the sync loop is required before a good sync state is achieved. The three computers begin the run state within the normal sync tolerance even though computer B is late.

In the original design, rescheduling of the minor cycle interrupt followed the crosslink job. Because of normal variations in the input-output execution times, synchronization discrete skew of approximately 40 microseconds was observed. When the interrupt scheduler was placed before the crosslink, a more stable sync process resulted. It was also found that because they were no longer looking for the third computer, the computers in the dual computer configuration were completing the synchronization job too quickly. The sync timing had to be adjusted to prevent interference with the input-output process. In addition, it was discovered that execution of a ground test display program in one computer resulted in a condition sufficiently out of sync to cause that channel to be declared failed. Changes in the display program and the sync algorithm software were required. The present algorithm is reasonably tolerant of transient disturbances, but still able to identify a permanent sync failure in less than 200 milliseconds. The achievement of synchronization and successful crosslink was found to be a strong positive indication of system health.



Ground tests have demonstrated the system's capability to resynchronize after numerous combinations of power shutdown and recovery. Following power application, the triplex system returns to the run state without ground crew or pilot interaction. Figure 24 illustrates a power shutdown and recovery sequence. Each of the three outputs of the pitch digital-to-analog converters (DAC's) is approximately 0.6 volts before the power is turned off. When channel A is repowered, it loops in the sync routine waiting for a second machine. The DAC voltage peak is due to the characteristics of the DAC hardware following initial power application. Within 200 milliseconds of the channel B power application, synchronization is achieved between channels A and B and the previous DAC output is restored. Likewise, channel C synchronizes with channels A and B after its power is restored.

### 5.2.2 Restart Performance

Critical to the normal turn-on sequence and to abnormal power activity is the operation of the restart routines that restore a channel to normal operation. The primary system was designed to be tolerant of transient interruptions in processing due to supply voltage faults or intermittent hardware faults. The primary design for restarts specified that DAC outputs be restored following the restart. Figure 25 shows the typical restart performance of the system. The three pitch DAC signals are initially approximately 1.2 volts. A series of power interruptions in channel A or channel B has no effect on the remaining two channels. In addition, each time power is restored to a channel, the channel performs a restart and is initialized to the current output state by the remaining two channels. The rate of restarts in the example is not high enough to hard fail any channel.

In the development of the restart process, some problems were encountered in assuring that a restarted computer had valid data before it resumed normal operation and that the other two channels remained unaffected by the process. Undesirable transients were observed on more than one DAC following a single channel restart. Modifications were made to the logic that selected the channel to be used as a source of data for the other channels, and to the sensor RM logic, which verified the presence of good data as indicated by a leading valid bit in each offset binary data word.

All features of the error handling routines, including hard-, soft-, and self-failure actions, were tested and found to perform as required, although the number of allowable provisional failures had to be increased from initial design values.

## 5.3 Control System Test Experience

### 5.3.1 Control Law Performance

Operation of the control system modes on the iron bird simulator has been very similar to that predicted during the analytical design phase. All modes have been evaluated over the flight envelope in closed-loop simulation. Pilot comments have been favorable, but only actual flight tests will reveal the extent of the benefits provided by many of the control modes, such as the ride smoothing and maneuver flap systems.

A substantial improvement in the airplane response is expected in the augmented modes. Figure 26 shows the response of the simulated F-8C aircraft to a pilot step input in the DIRECT, SAS, and CAS modes. The short period response is markedly improved in the augmented modes. Figure 27 shows the Dutch roll response of the airplane in the DIRECT and SAS modes in the roll and yaw axes. Dutch roll damping is improved in the SAS modes. Other results have shown that sideslip due to aileron inputs is reduced in the SAS modes.

The operation of the angle-of-attack limiter during piloted simulation is shown in figure 28. The limit for angle of attack was set at  $18^\circ$ . During the first part of the maneuver (a wind-up turn), the limit angle of attack is reached and held within  $1^\circ$  while the pilot smoothly moves the stick full back. In the second part of the maneuver, a rapid full back stick maneuver is demonstrated. As in the first part of the maneuver, angle of attack is held within  $1^\circ$  of the limit until back stick pressure is released sufficiently to return control to the pilot.

No significant problems were encountered in the development and refinement of the control laws on the iron bird. Some adjustments to the gain schedules were necessary to improve performance at some high angle-of-attack and high dynamic pressure conditions.

### 5.3.2 Actuator Performance

The design bandwidth and hysteresis for three-channel operation have been met. However, some developmental problems were encountered in the pressure equalization circuit. Two of the hydraulic systems used to power the secondary actuators were also used to supply the aircraft's power actuators. Because the third hydraulic system served only utility functions, during surface movement it did not display as large a pressure drop as the other two systems. One result of this was a small actuator oscillation during quiescent operation. This problem was alleviated by adjustments in the pressure equalization bandwidth and deadband.

Fluctuations in pressure in the channel supplied by the utility hydraulic system were also smaller than those of the other two channels when an actuator was operated into its stops, and, as a result, were likely to cause nuisance disconnects of that channel. To prevent operation into and out of the secondary actuator stops, the position command to the actuator was limited.

One additional problem that required system modification was the skew in the turnoff command for the two remaining actuation channels after two hardover failures in one actuator. The differential pressure existing during the 50- to 100-millisecond skew was sufficient to drive the control surface several degrees before the actuator was completely disabled.

## 6.0 CONCLUDING REMARKS

A flight-critical, fault-tolerant digital fly-by-wire primary control system has been designed, developed, and ground tested in an iron bird facility and is near flight maturity. Ground tests have demonstrated the system's ability to continue operation during normal system variations and to resume operation after transient

faults. To sufficiently test the sensor and discrete redundancy management algorithms, a special real-time automated test sequence was developed, which substantially reduced the engineering test time required.

The majority of the developmental work on the system was required in the area of computer redundancy management where the three channels were required to function as three independent devices instead of as one. Good synchronization performance has been achieved and allows a sensitive determination of system health. Moderate changes were made in the computer redundancy management algorithms during the ground test phase to eliminate common-mode faults.

The sensor redundancy management software requires a large amount of execution time relative to the control law computation, but less memory.

Ground simulation has shown that the control laws provide the predicted improvement in the aircraft's response and handling qualities. The results of the flight test program, however, will ultimately determine the overall acceptability of the system.

#### REFERENCES

1. Deets, D. A.; and Szalai, K. J.: Design and Flight Experience With a Digital Fly-by-Wire Control System Using Apollo Guidance System Hardware on an F-8 Aircraft. AIAA Paper 72-881, Aug. 1972.
2. Deets, Dwain A.; and Szalai, Kenneth J.: Design and Flight Experience With a Digital Fly-by-Wire Control System in an F-8 Airplane. Advances in Control Systems, AGARD-CP-137, May 1974, pp. 21-1-21-10.
3. Description and Flight Test Results of the NASA F-8 Digital Fly-by-Wire Control System. NASA TN D-7843, 1975.
4. Szalai, Kenneth J.; and Deets, Dwain A.: F-8 Digital Fly-by-Wire Flight Test Results Viewed From an Active Controls Perspective. Impact of Active Control Technology on Airplane Design, AGARD-CP-157, June 1975, pp. 22-1-22-14.
5. Eggers, James A.; and Bryant, William F., Jr.: Flying Qualities Evaluation of the YF-16 Prototype Lightweight Fighter. AFFTC-TR-75-15, Air Force Flight Test Center, Edwards Air Force Base, July 1975.
6. Sutton, M. L.; Hasson, W. J.; and Soderlund, G. M.: Feasibility Study for an Advanced Digital Flight Control System (DIGIFLIC). Volume I: Summary, Analysis, and System Studies. ADR-773, Lear Siegler, Inc., Oct. 1972.
7. Borow, M. S.; Gaabo, R. J.; Hendrick, R. C.; Konar, A. F.; Lahn, T. G.; Markusen, D. L.; Smith, F. L.; Schmitz, H. G.; and Sowada, D. J.: Navy Digital Flight Control System Development. 21857-FR, Honeywell Inc., Dec. 1972.
8. Honeywell Inc.: Flight Test Evaluation of a Digital Multimode Flight Control System for the A-7D Aircraft. AFFDL-TR-75-97, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Aug. 1975. Volume I - Design, Analysis, and Ground Test Phase. Volume II - Flight Test.
9. McDonnell Aircraft Company: Definition Study for an Advanced Fighter Digital Flight Control System. AFFDL-TR-75-59, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, June 1975.
10. Fairbanks, L. E.; and Templeman, J. E.: Flight-Critical Digital Control System Evaluation. AIAA Paper 75-566, Apr. 1975.
11. Osder, S. S.; Massman, D. C.; and Devlin, B. T.: Flight Test of a Digital Guidance and Control System in a DC-10 Aircraft. AIAA Paper 75-567, Apr. 1975.
12. Hendrick, R. C.; and Hill, C. D.: Self-Testing Digital Flight Control Applications. AIAA Paper 75-568, Apr. 1975.
13. Yopp, W. B.; and McDonnell, J. D.: Digital Flight Control Systems - Considerations in Implementation and Acceptance. AIAA Paper 75-577, Apr. 1975.
14. Kestek, Richard E.: YC-14 Digital Flight Control Data Management. AIAA Paper 75-1087, Aug. 1975.
15. Reeder, John P.; Taylor Robert T.; and Walsh, Thomas M.: New Design and Operating Techniques and Requirements for Improved Aircraft Terminal Area Operations. NASA TM X-72006, 1975.
16. Klass, Philip J.: Airborne Computer Design Concept Shifts. Aviation Week & Space Technology, vol. 102, no. 7, Feb. 1975, pp. 53-55.
17. Mulcare, D. B.; and Benson, J. W.: Balanced Functional Design of Automatic Digital Flight Control Systems. AIAA Paper 75-1088, Aug. 1975.
18. Kubbat, Wolfgang J.: A Quadruredundant Digital Flight Control System for CCV Application. Impact of Active Control Technology on Airplane Design, AGARD-CP-157, June 1975, pp. 15-1-15-9.
19. O'Hern, Eugene A.: Space Shuttle Avionics Redundancy Management. AIAA Paper 75-571, Apr. 1975.
20. Hartmann, Gary L.; Hauge, James A.; and Hendrick, Russell C.: F-8C Digital CCV Flight Control Laws. NASA CR-2629, 1976.
21. Edwards, John W.: A FORTRAN Program for the Analysis of Linear Continuous and Sampled-Data Systems. NASA TM X-56038, 1976.

TABLE 1.—INPUT SENSORS

Sensor	Redundancy level	Signal type
Pitch rate . . . . .	3	dc
Roll rate . . . . .	3	dc
Yaw rate . . . . .	3	dc
Axial acceleration . . . . .	3	dc
Lateral acceleration . . . . .	3	dc
Normal acceleration . . . . .	3	dc
Pitch center stick position . . . . .	3	ac LVDT
Roll center stick position . . . . .	3	ac LVDT
Pitch side stick force . . . . .	3	ac LVDT
Roll side stick force . . . . .	3	ac LVDT
Rudder pedal position . . . . .	3	ac LVDT
Angle of attack . . . . .	2	dc
Angle of sideslip . . . . .	1	dc
Horizontal stabilizer actuator position . . . . .	3	dc potentiometer
Surface positions (5) . . . . .	1	dc potentiometer
Pitch attitude . . . . .	2	Synchro
Roll attitude . . . . .	2	Synchro
Heading angle . . . . .	2	Synchro
Wing position . . . . .	3	dc potentiometer
Mach number . . . . .	2	dc potentiometer
Altitude . . . . .	2	Serial digital
Computer temperature . . . . .	1	dc

TABLE 2.—INPUT DISCRETES

Discrete	Redundancy level
Mode select . . . . .	3
Autopilot panel . . . . .	2
Annunciator reset . . . . .	2
Trim . . . . .	2
Sidestick enable . . . . .	3
CIP digits . . . . .	1
CIP enter/clear . . . . .	2
Wing up/down . . . . .	3
Weight on wheels . . . . .	1
Gain switches . . . . .	1
Gear position . . . . .	3
Autopilot disconnect . . . . .	1
Computer bypass status . . . . .	3

TABLE 3.—ACTUATOR CHARACTERISTICS

	Bandwidth, rad/sec	Slew rate	Hysteresis, percent of full stroke	Force or torque output	Stroke
Secondary actuators	125.0	30 cm/sec	0.2	10,000 N	±2.5 cm
Power actuators —					
Pitch	12.5	25 deg/sec	0.1	3,000 N-m	+6.75°, -26.5°
Roll	30.0	70 deg/sec	0.1	1,080 N-m	+45°, -15°
Yaw	25.0	120 deg/sec	0.1	356 N-m	±21°

TABLE 4.—DIGITAL CONTROL COMPUTER CHARACTERISTICS

Central processor unit —	
Number system . . . . .	Binary, fixed point, two's complement, fractional hexadecimal floating point
Operation . . . . .	Full parallel
Fixed-point data . . . . .	16 and 32 bits, including sign
Floating-point data . . . . .	32 bits (24-bit mantissa) and 64 bits (56-bit mantissa)
Typical execution times, register to register, $\mu\text{sec}$ —	
Fixed point:	
Addition . . . . .	1.0
Multiplication . . . . .	4.8
Division . . . . .	8.4
Floating point:	
Addition . . . . .	2.4
Multiplication . . . . .	5.0
Division . . . . .	10.5
Average instruction rate, per sec . . . . .	480,000
Main storage —	
Type . . . . .	Random access, destructive readout, ferrite magnetic core, nonvolatile
Cycle time, nsec . . . . .	900 read/write
Addressable unit . . . . .	16-bit halfword
Features . . . . .	Parity and store protect on halfword
Input-output —	
Type . . . . .	Parallel halfword plus parity, multiplex, half duplex
Maximum data rate, per sec . . . . .	225,000 full words
Discretes . . . . .	Four input, four output
External interrupts . . . . .	Five levels
Physical characteristics —	
Weight, kg . . . . .	24 (32,000-word memory)
Volume, $\text{m}^3$ . . . . .	0.025
Power, W . . . . .	375 (32,000-word memory)
Environment . . . . .	MIL-E-5400 Class 2X

TABLE 5.—SOFTWARE MEMORY ALLOCATION

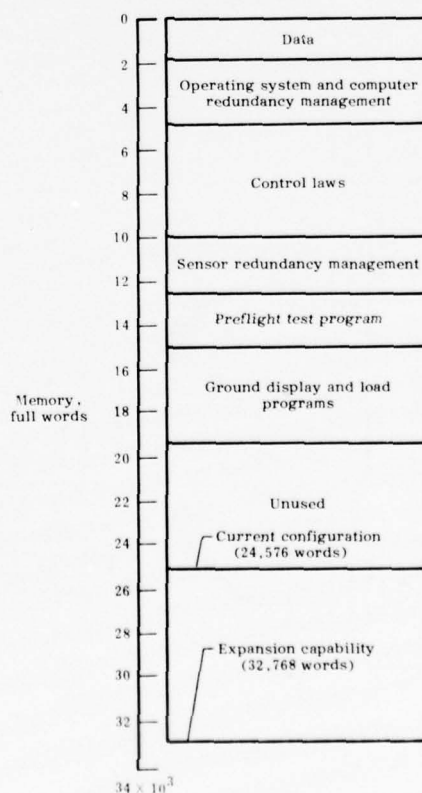




TABLE 6.—PRIMARY-TO-CBS SWITCHING LOGIC

Event	Switch state		
	Switch A	Switch B	Switch C
Comparator CA reports failure	CBS	PRI	PRI
Comparators CA and CB report failure	CBS	CBS	CBS
Channel C digital failure	PRI	PRI	CBS
Channels B and C digital failure	CBS	CBS	CBS
Pilot selects CBS	CBS	CBS	CBS

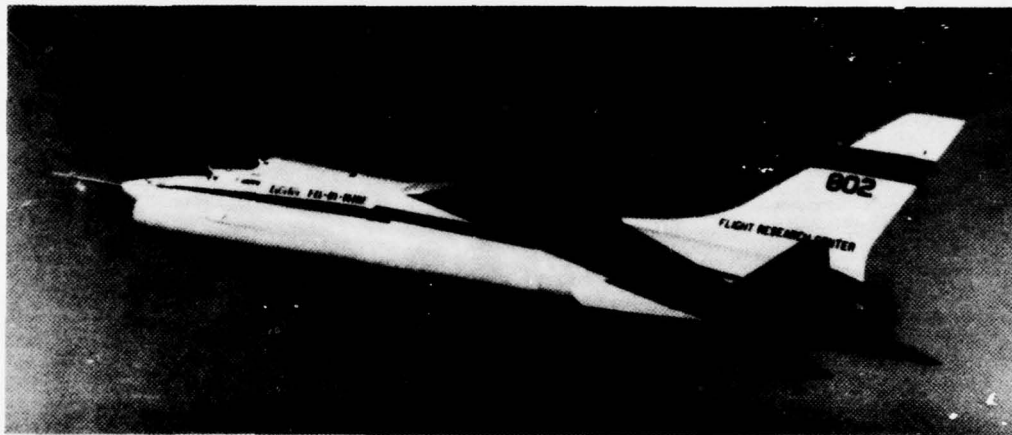


Figure 1. F-8 digital fly-by-wire aircraft.

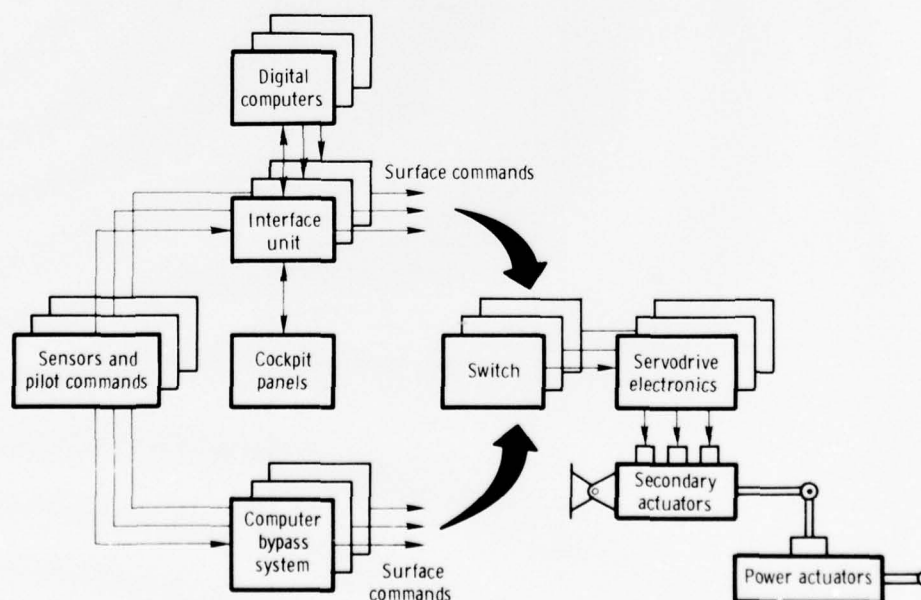


Figure 2. F-8 DFBW control system mechanization.

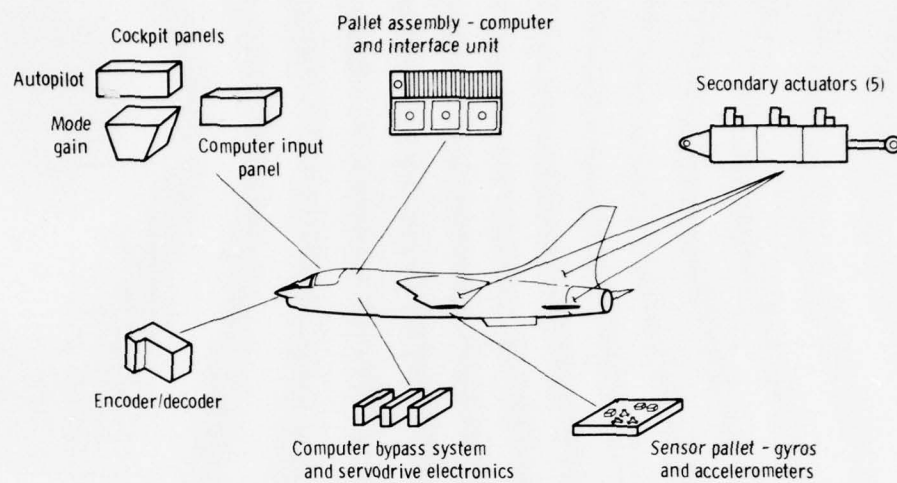


Figure 3. F-8 DFBW hardware elements.

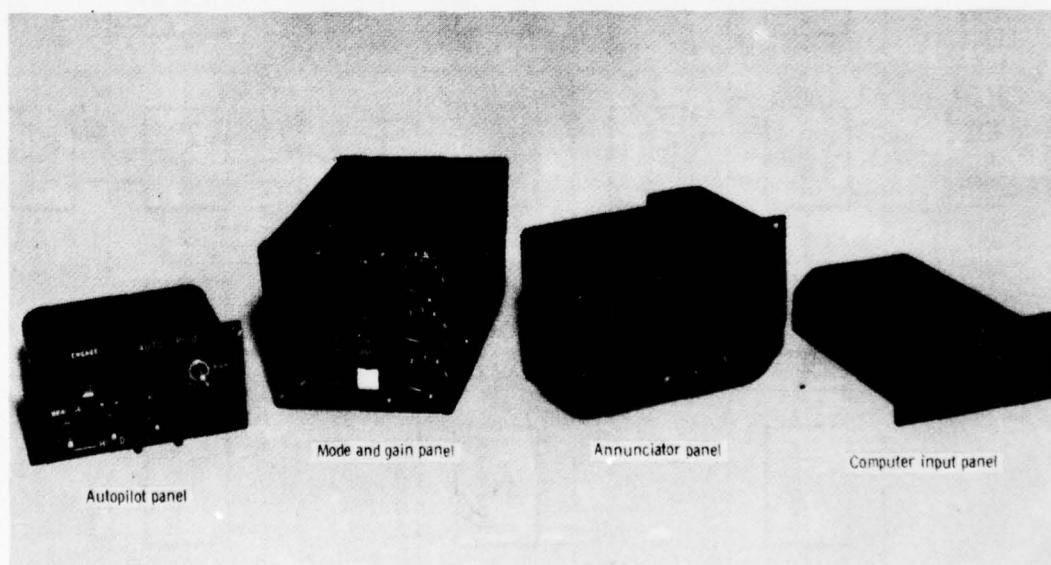


Figure 4. Cockpit panels.

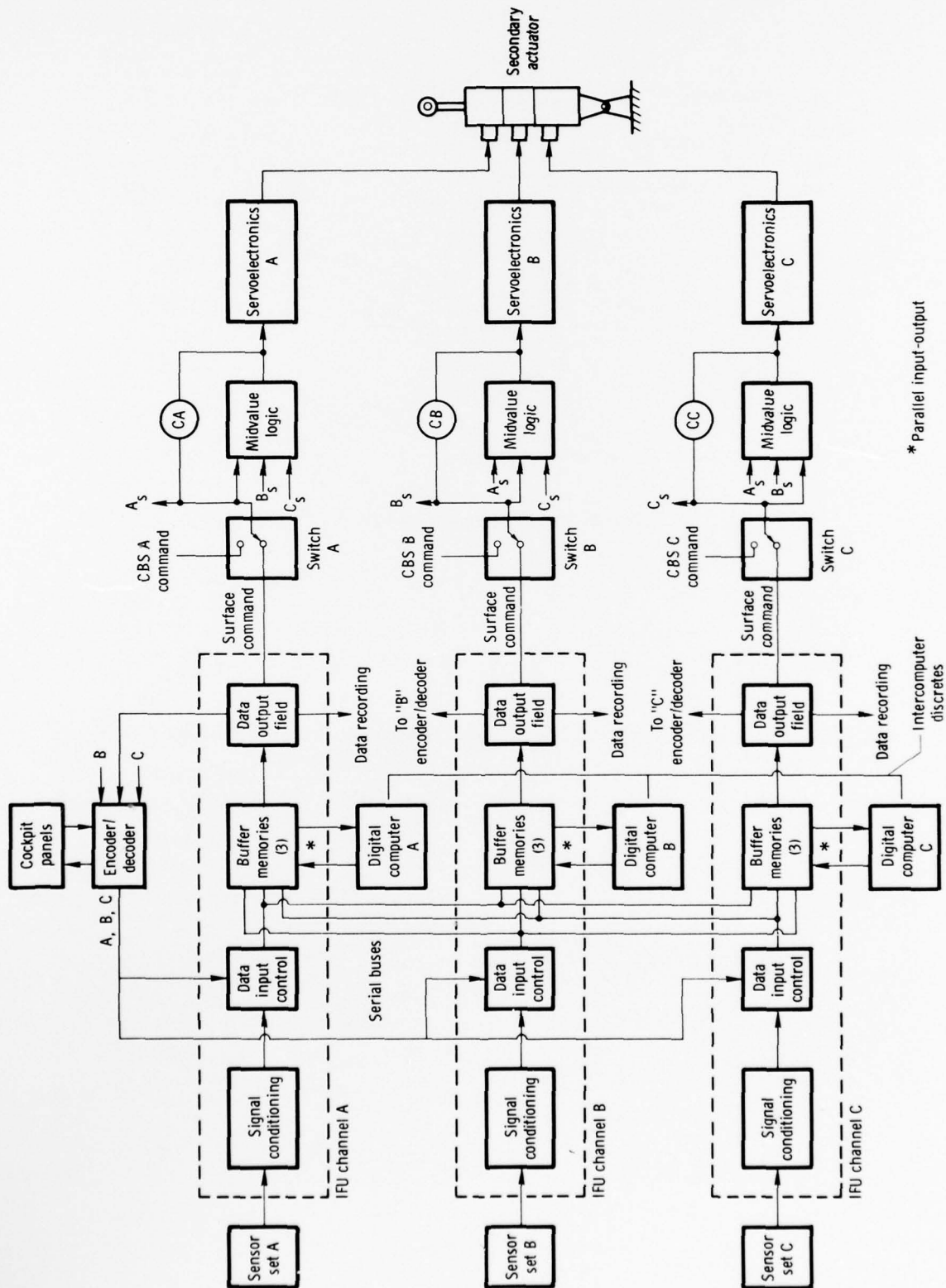


Figure 5. Primary digital system mechanization.

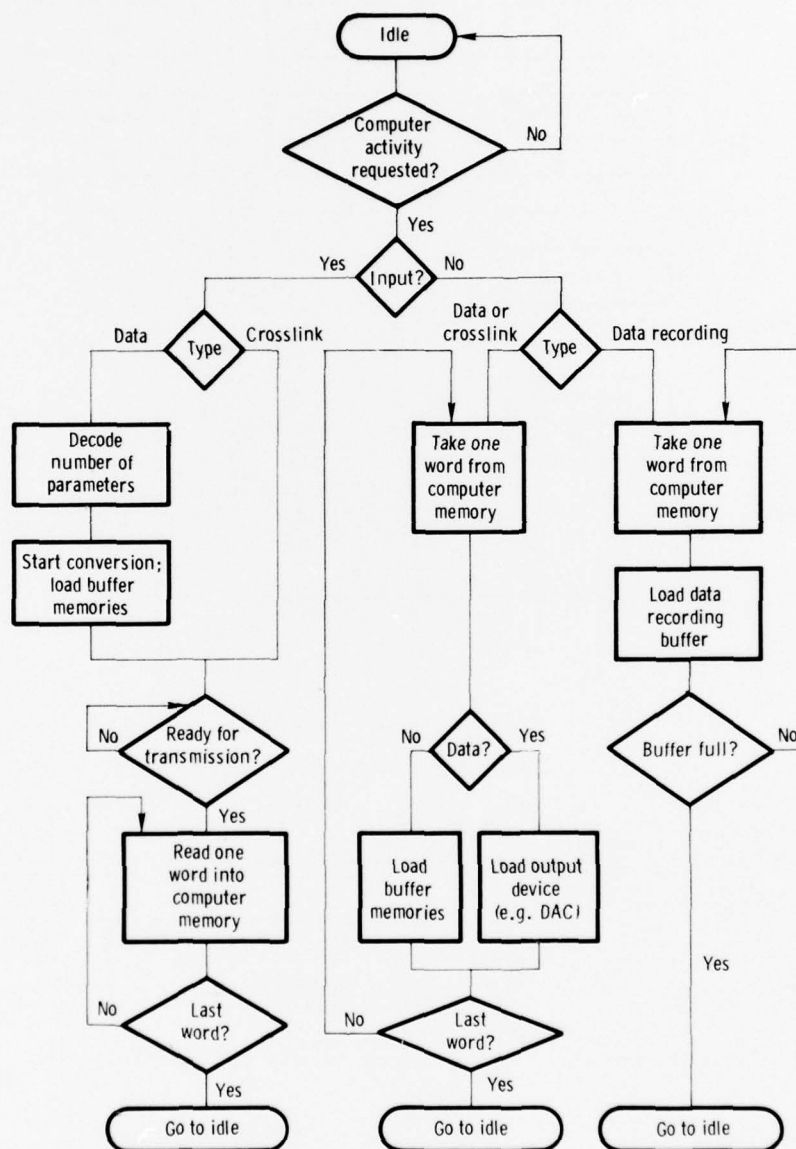


Figure 6. Interface unit microprogram flow.



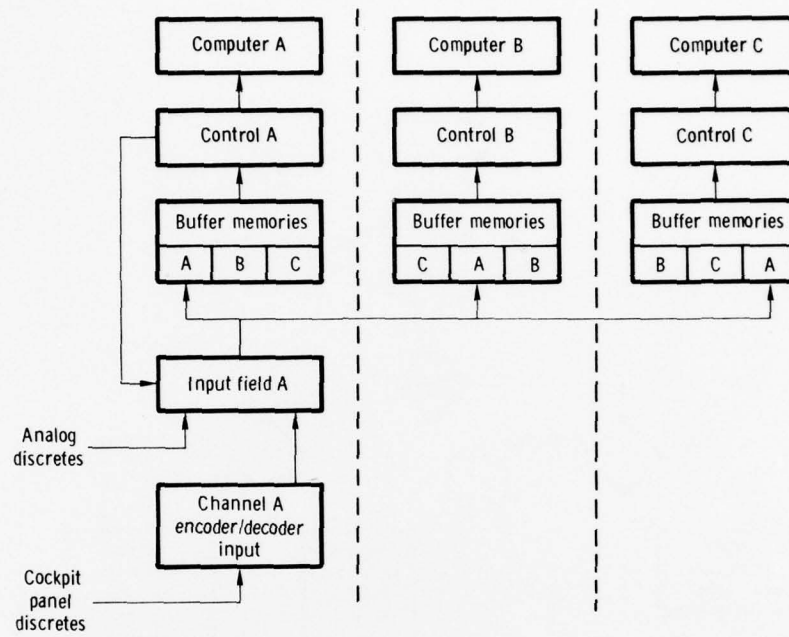


Figure 7. Input data flow for channel A.

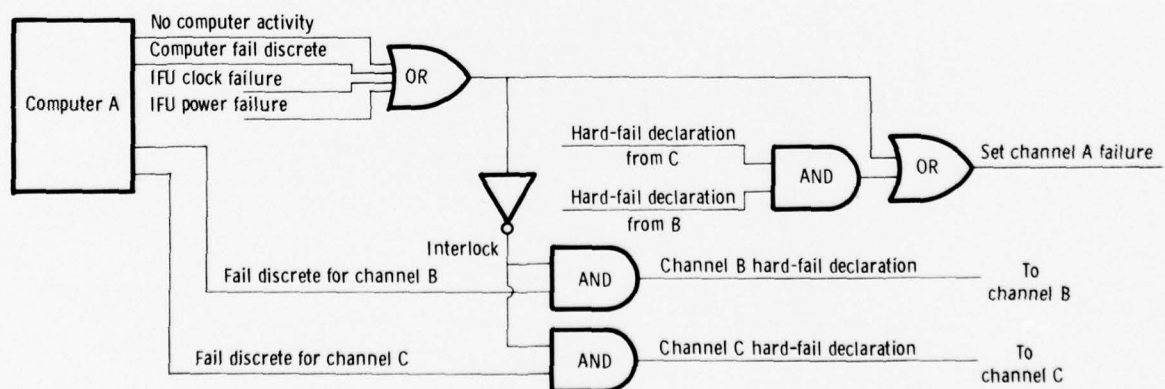


Figure 8. Functional diagram of fault detection hardware in IFU (shown for channel A).

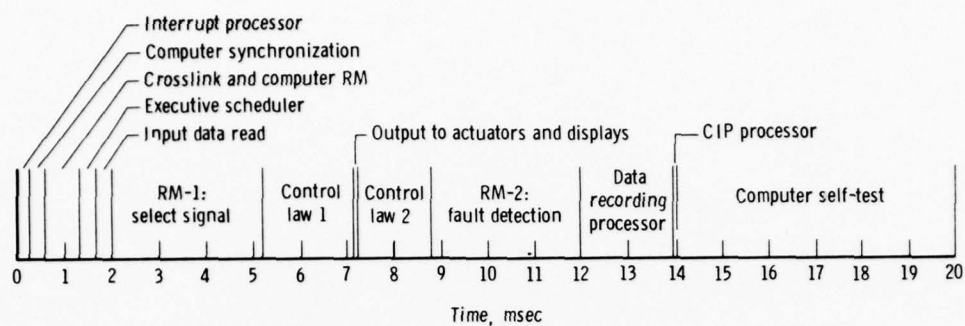


Figure 9. Software sequence and timing during one minor cycle. Three channels, direct modes.

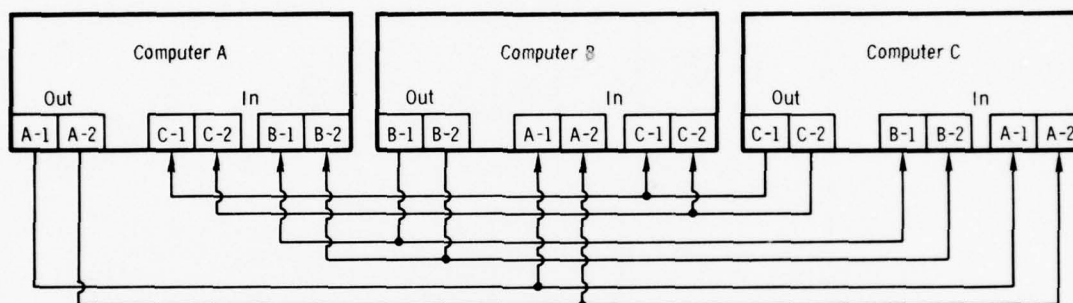


Figure 10. Synchronization discretes.

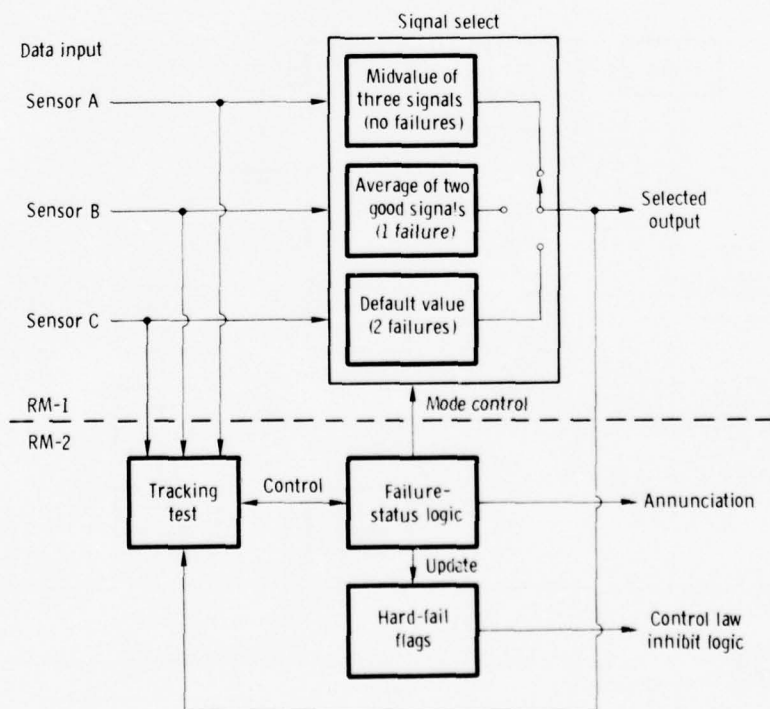


Figure 11. Triplex analog sensor redundancy management algorithm.

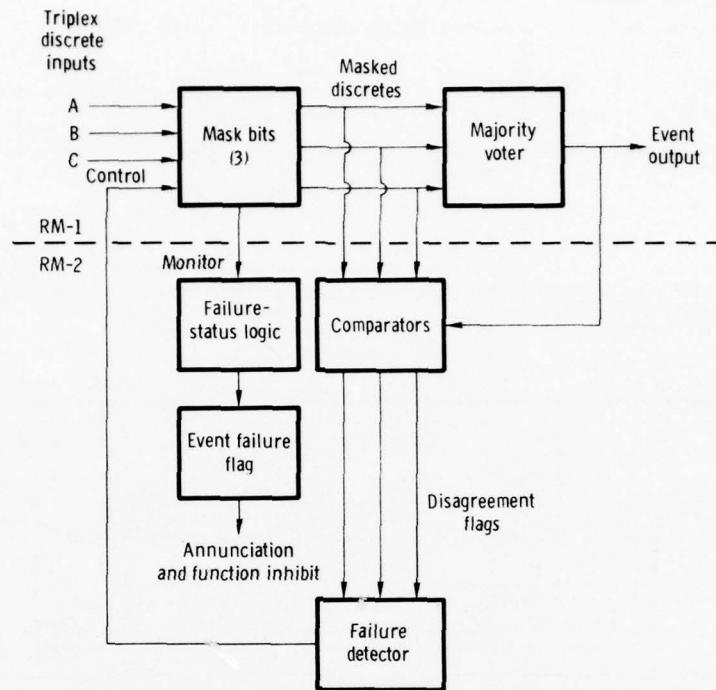


Figure 12. Triplex discrete RM algorithm.

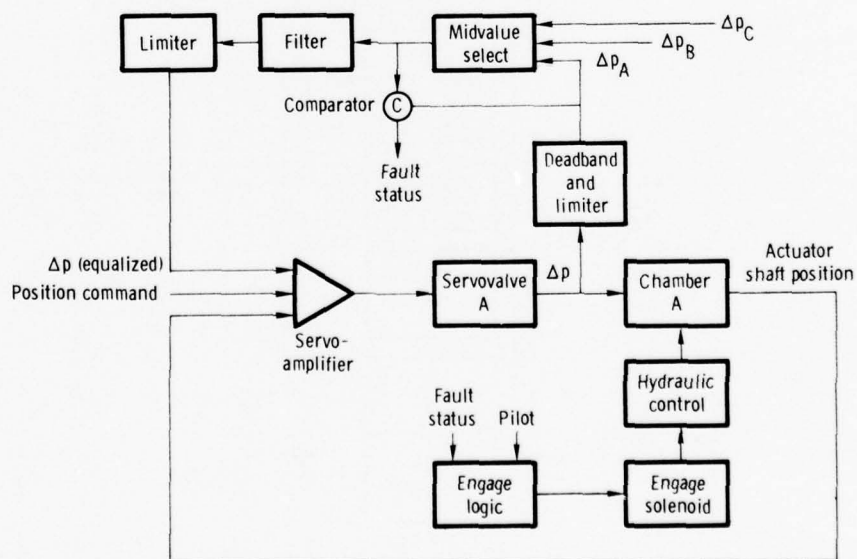


Figure 13. Schematic diagram of single channel of secondary actuator and servoelectronics.

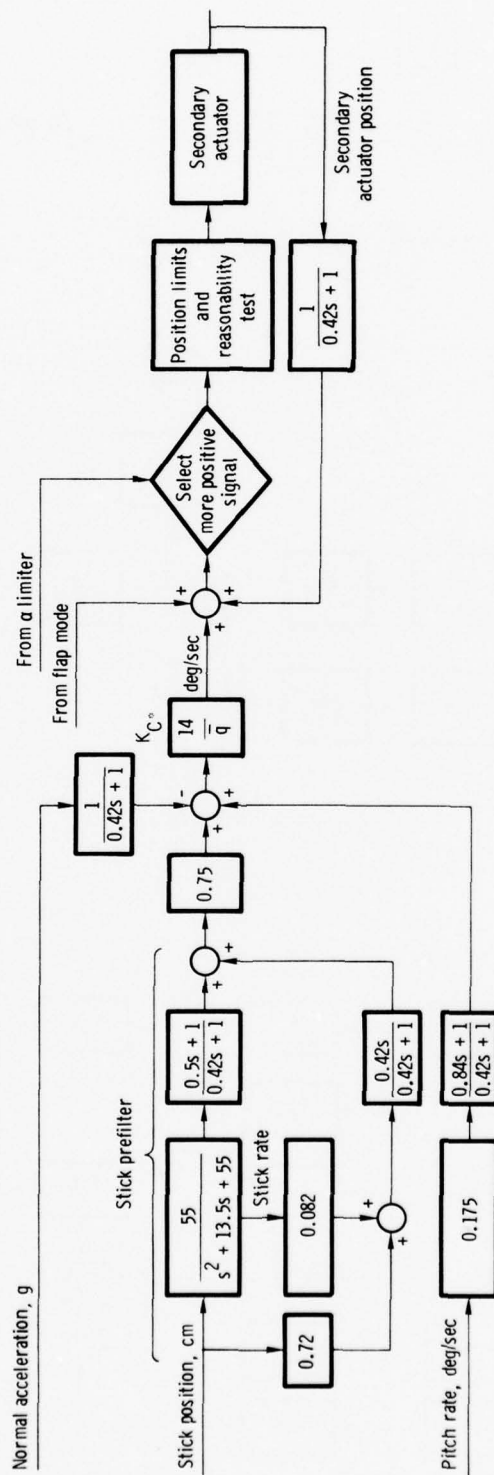


Figure 14. Pitch CAS control system.



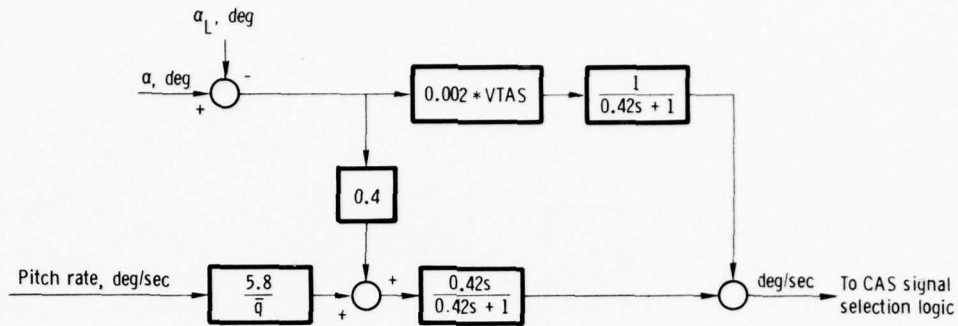


Figure 15. Angle-of-attack limiter.

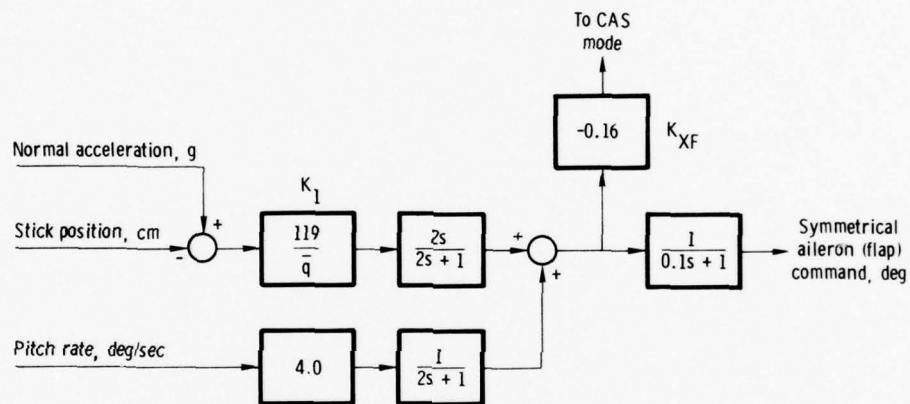


Figure 16. Direct lift mode.

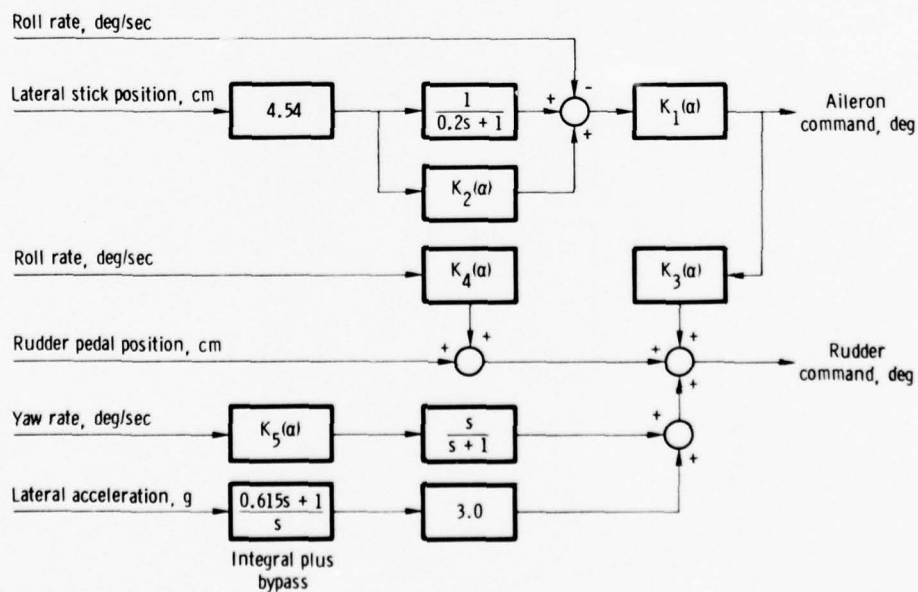


Figure 17. Lateral-directional SAS modes.

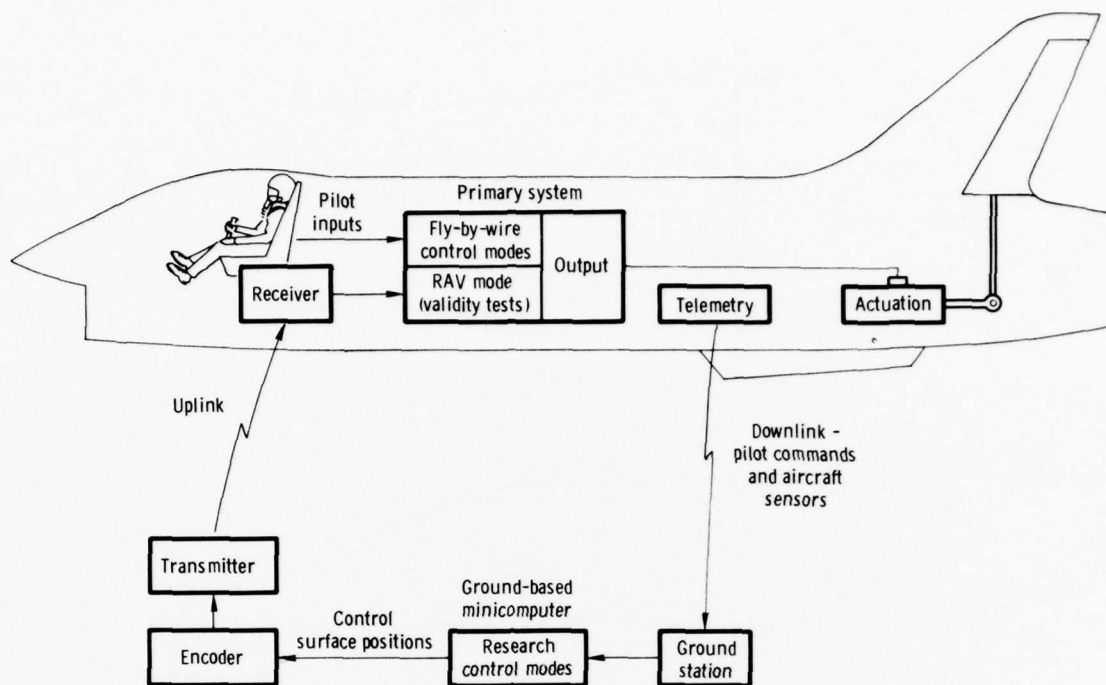


Figure 18. F-8 DFBW remote augmentation vehicle (RAV) concept.

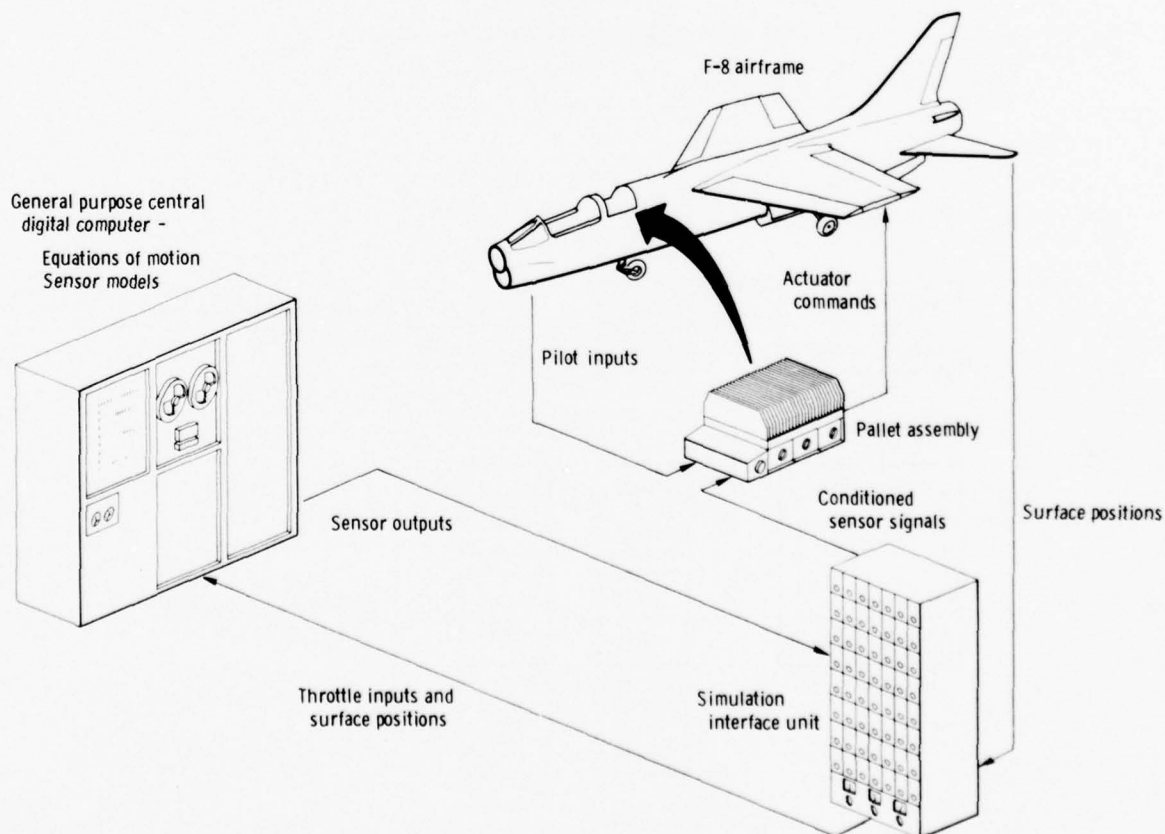


Figure 19. Iron bird facility.

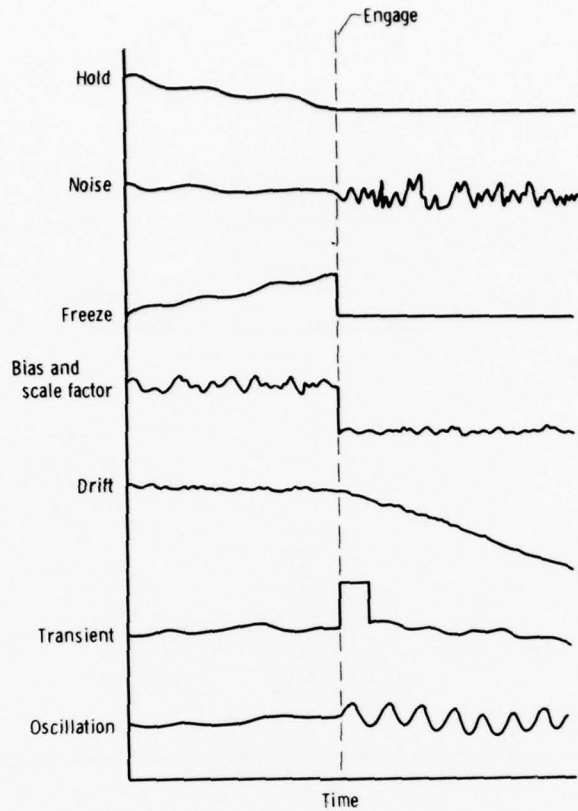


Figure 20. Simulated sensor wave forms.

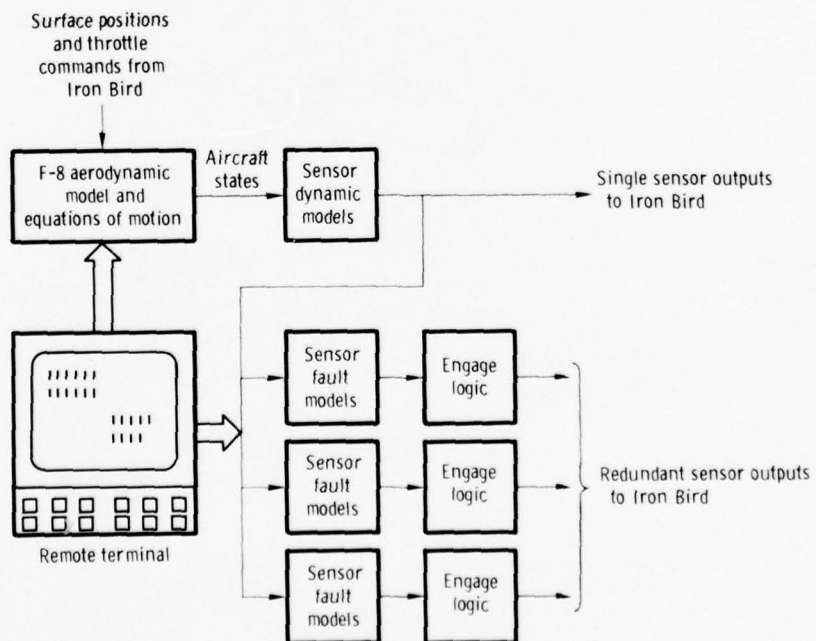


Figure 21. Integration of redundant sensor models in real-time simulation.

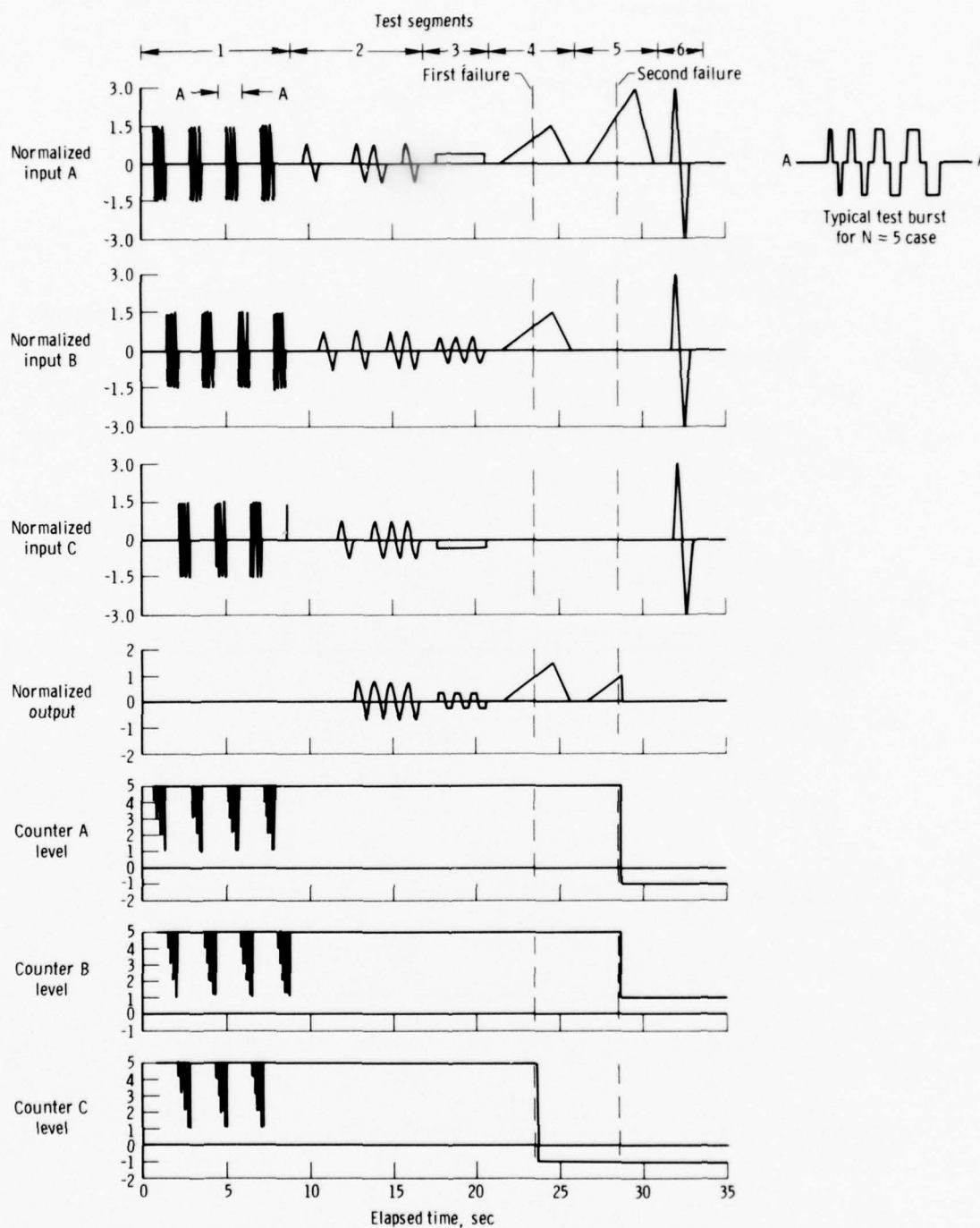


Figure 22. Sample sensor test results for  $N = 5$  case.



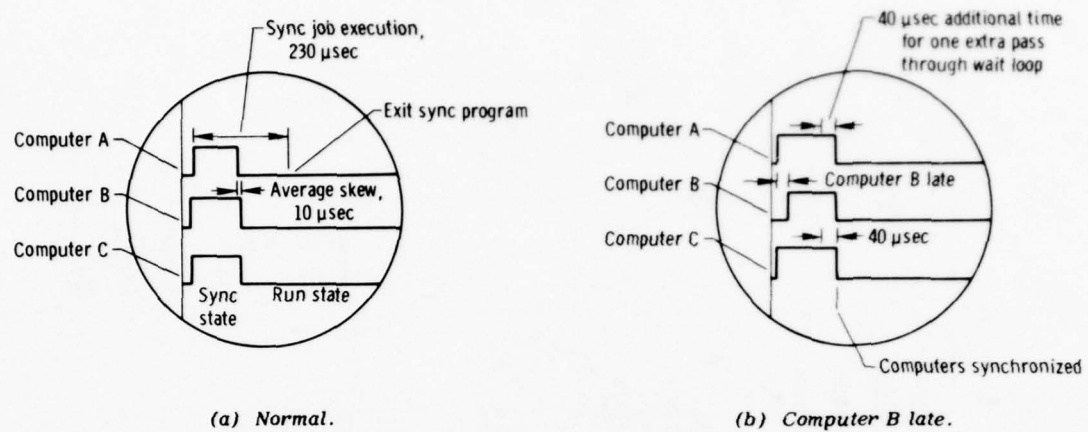


Figure 23. Synchronization performance.

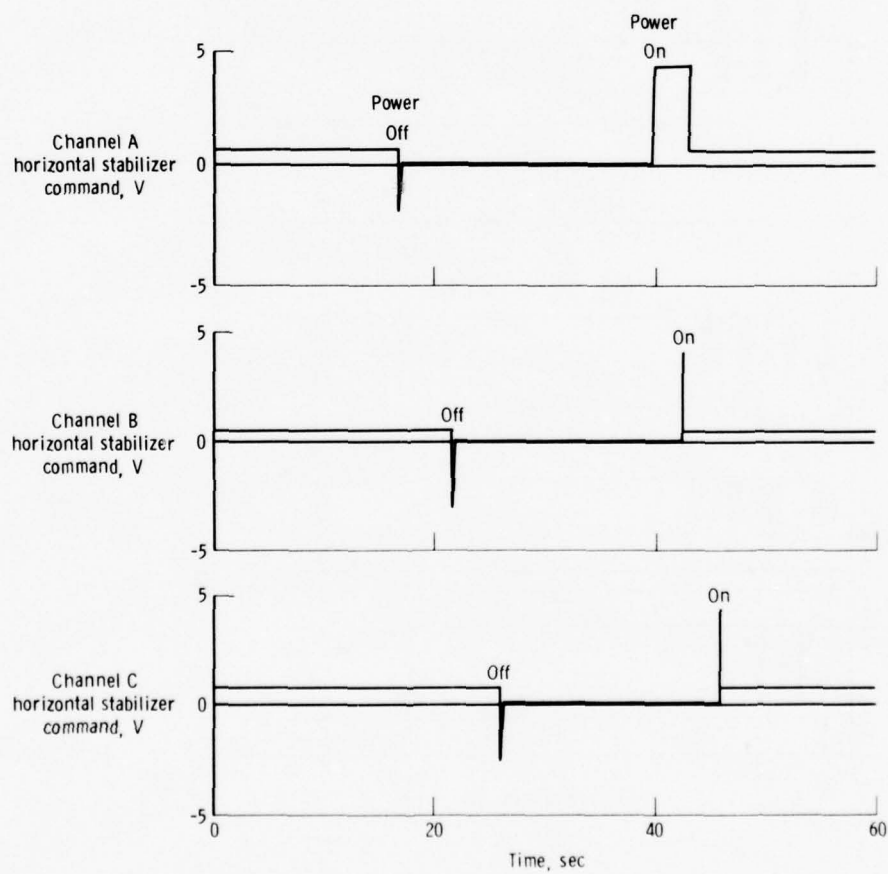


Figure 24. Power shutdown and recovery sequence.

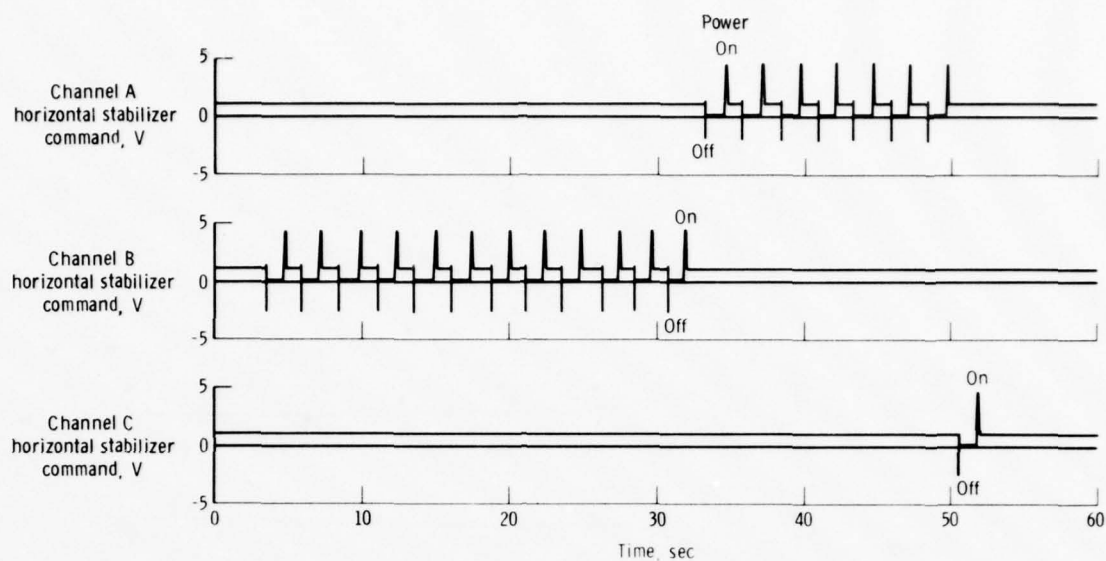
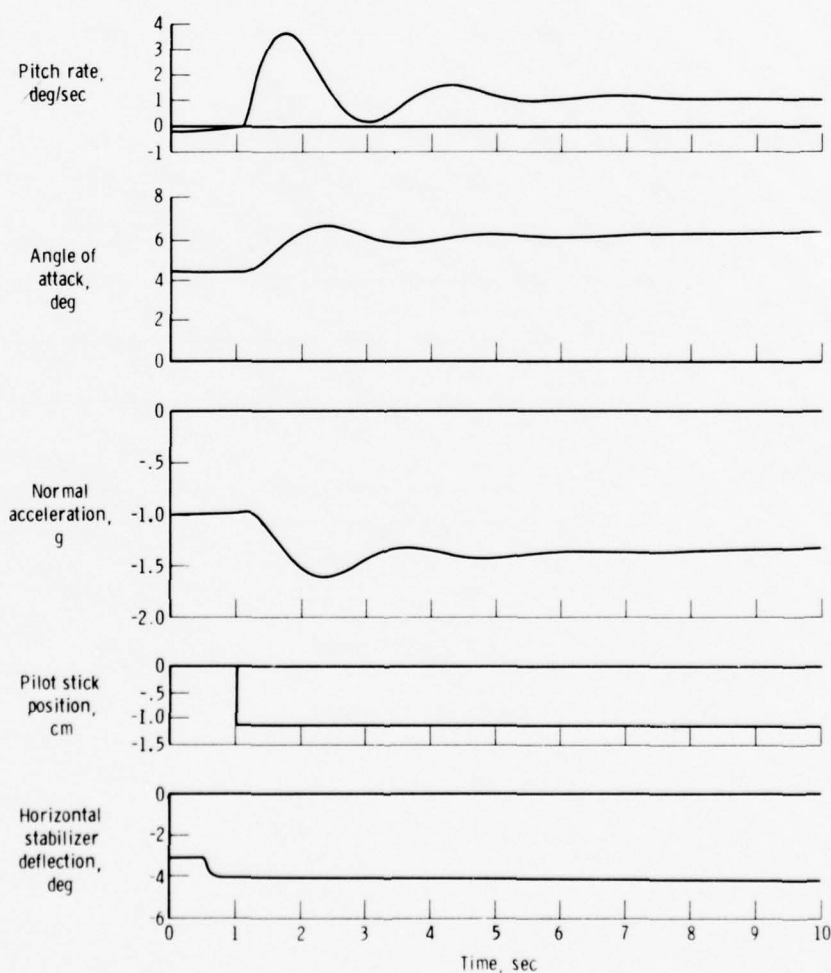
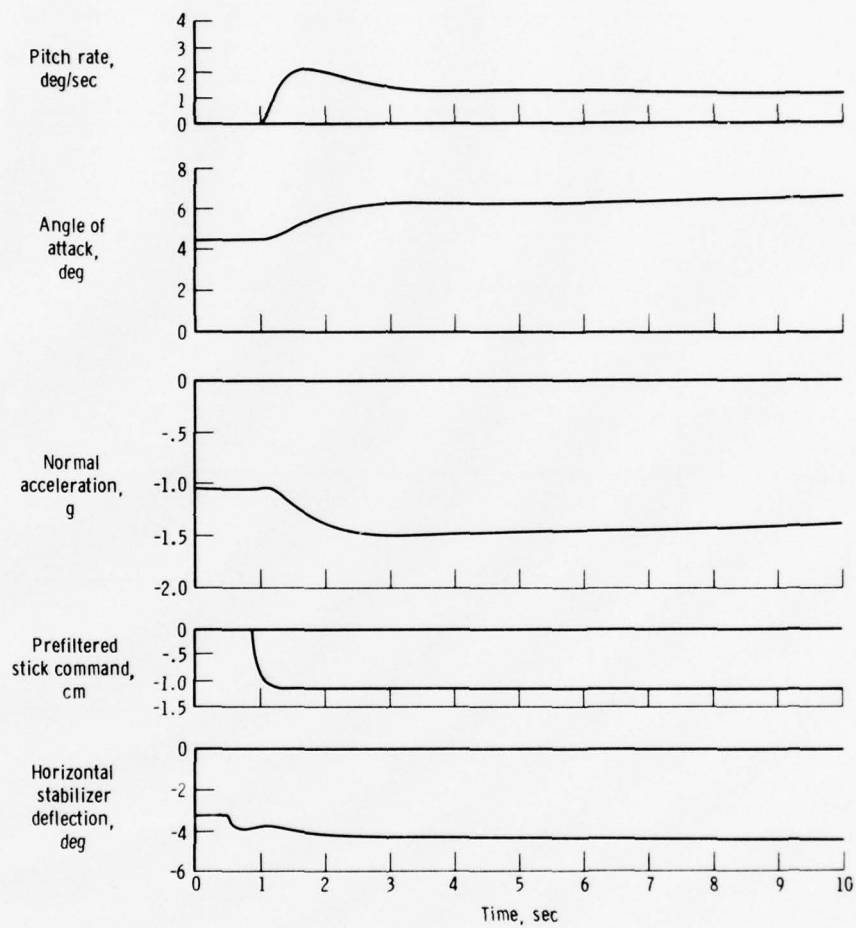


Figure 25. Restarts induced by interruptions in power.



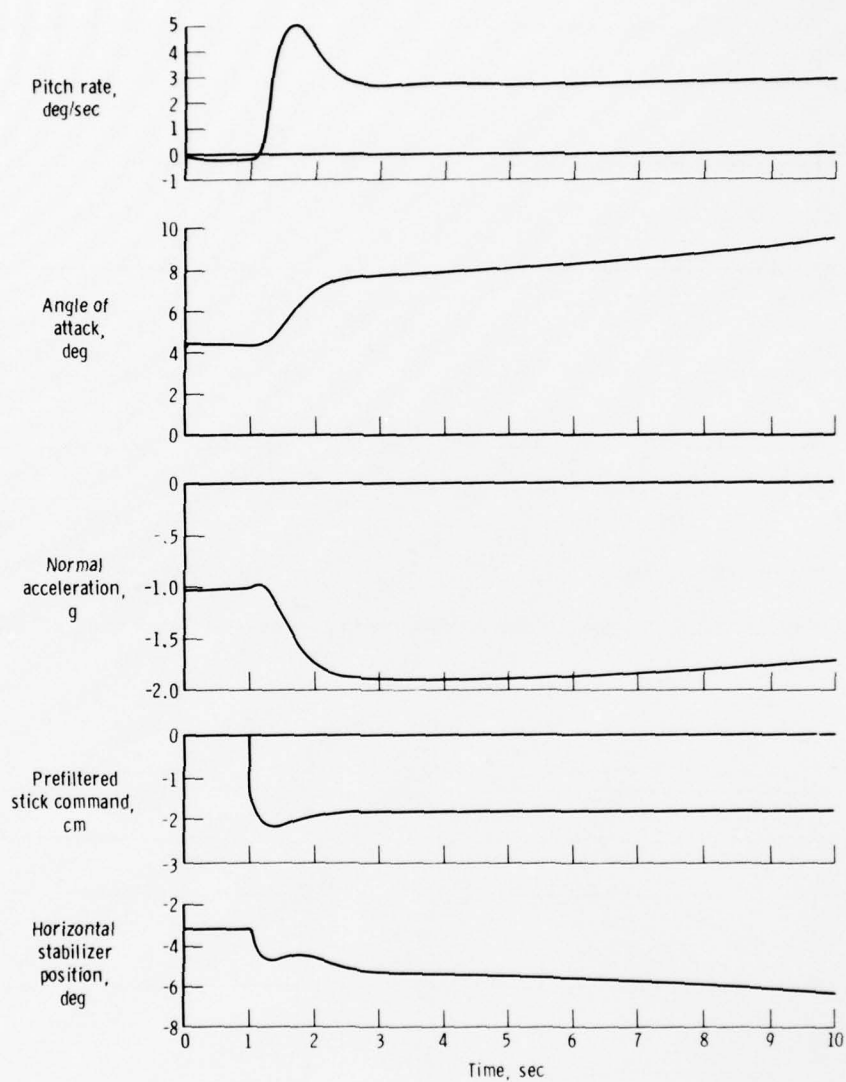
(a) DIRECT mode.

Figure 26. Longitudinal response of the simulated F-8 DFBW aircraft. Mach 0.6; altitude, 6100 m.



(b) Pitch SAS mode.

Figure 26. Continued.



(c) Pitch CAS mode.

Figure 26. Concluded.



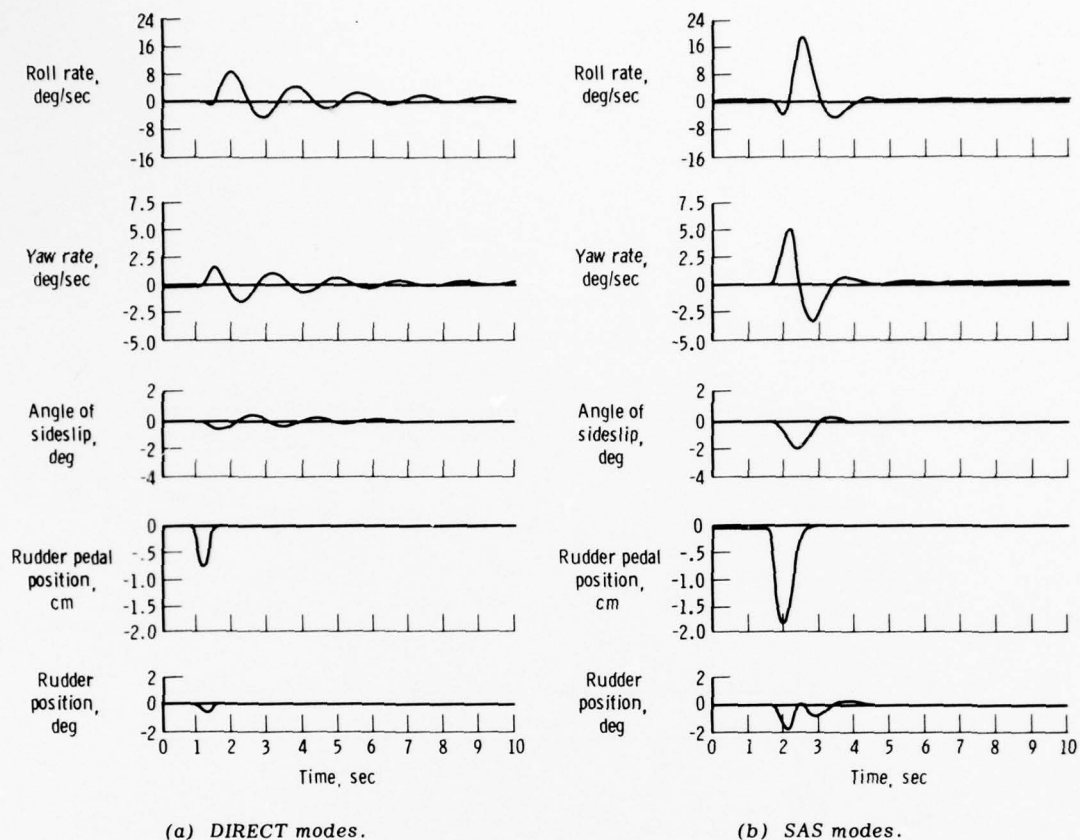


Figure 27. Lateral-directional response of the simulated F-8 DFBW aircraft. Mach 0.8; altitude, 6100 m.

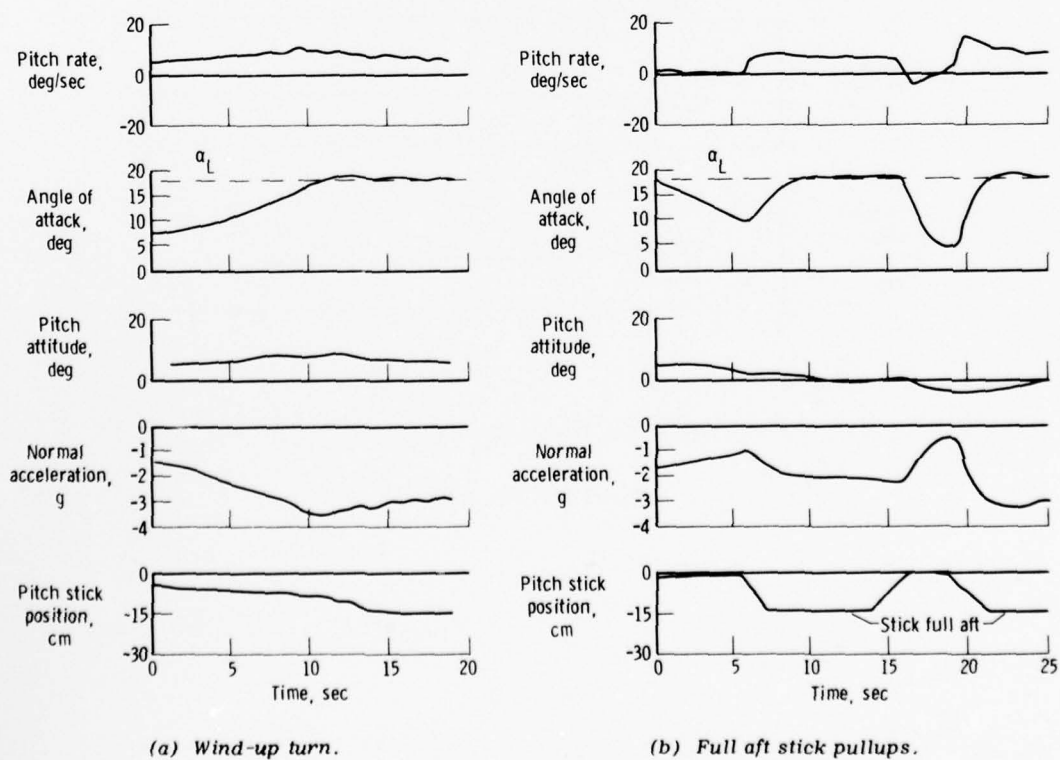


Figure 28. Performance of angle-of-attack limiter.

## L-1011 FLIGHT CONTROL SYSTEM

by

J.A. Flapper, Research and Development Engineer  
E.O. Throndsen, Research and Development Engineer  
Lockheed-California Company  
P.O. Box 551  
Burbank, California

### SUMMARY

This chapter describes those aspects of the L-1011 flight controls - primary and automatic - which are of interest because of the state of the art advancements and the improvements which they represent. The flying tail primary control system, its rationale and design features is dealt with in some depth. Integration of primary controls with the automatic flight controls is treated and the direct lift control system and roll control briefly described. The automatic controls are described with emphasis on the yaw stability augmentation system and the automatic landing system. The former is in concept an "active control system" in that design loads are predicated on its availability. The latter, for the final stage of landing in Category III conditions, is the forerunner of the "fly-by-wire" concept for commercial transports. Again, system rationale or design features which enhance safety and reliability are treated.

### INTRODUCTION

The L-1011 TriStar, one of the current generation of wide-body commercial transports, is a short to medium range airplane that cruises typically at 33,000 feet and .85 Mach. Maximum takeoff and landing weights are around 470,000 and 360,000 pounds. Depending on accommodations, it can provide space for up to 400 passengers plus a crew of 13. Figure 1 gives the airplane dimensions.

The Tri-Star has been in airline operation since initial certification by the FAA in April 1972. It has been subsequently certificated by Canada (MOT), Great Britain (CAA), Japan (JCAB), West Germany (LBA), Saudi Arabia (DGCA) and Hong Kong (CAD). With over 130 airplanes presently in service, there have been about 700,000 revenue flight hours accumulated to date (4-1/2 years after initial certification) at a rate of approximately 1.9 hours per revenue flight.

Although the L-1011 does not extend its flight envelope beyond the limits of precedent commercial subsonic jet aircraft, the higher performance and safety demands plus the size of the aircraft and the availability of more advanced technology items provided an opportunity to take a fresh look at control system concepts. The result has been that many advances and improvements have been incorporated into both primary and automatic flight control systems in the TriStar. It is intended in the following discussion to describe some highlights of these systems and features which relate to improvements in safety, performance and overall reliability.

The Primary Flight Control section discusses some special requirements and then describes aspects of the system implementation which to a large extent are unique in the L-1011 flight control system. The pitch control system, being the most unconventional one, partly because of the flying tail concept, is treated in some detail. Discussed are the general system configuration and special features of the horizontal stabilizer power control system and the input system. The integration of the pilot input drive with the trim system is of particular interest. The pitch autopilot servo system is also discussed in some detail. In addition some features of the Direct Lift Control (DLC) system are described, with the emphasis on the DLC servo system. The roll and yaw control systems are only briefly treated.

The automatic flight control subsystems of the L-1011 Avionic Flight Control System are highly redundant in comparison to such systems of the previous generation of aircraft. This redundancy resulted from the need for high integrity in the Category III Automatic Landing System and the redundant configurations of the "cruise" portions of the autopilot and yaw control channels were also affected by this Category III requirement. An overview of the overall system arrangement and operational features is given along with the physical characteristics of the significant components. The overview is followed by a discussion of the principal features of the yaw stability augmentation system, the "cruise" autopilot and the automatic landing system. In each of these three sections design criteria and basic integrity related requirements are considered as well as the mechanization schemes and design features incorporated.

### PRIMARY FLIGHT CONTROLS

#### System Requirements

The present generation of wide-bodied airplanes has not materially expanded the flight envelope of the previous commercial jet fleet and in that respect the functional performance requirements for the L-1011 primary flight controls did not change significantly from those of previous commercial jets. However, the public safety demands which greatly increased over the last decade, partially as a result of a large surge in airplane travel and partially as the result of increased public consciousness of safety in transportation, demanded a significant improvement in system integrity. The increase in size of the new generation of airplanes and the resulting increase in potential magnitude of accident effects served only to emphasize the need for speedy definition of new standards and for their immediate implementation. Accordingly the U.S. Federal Aviation Agency (FAA) initially issued "Special Conditions" which were to be considered as an integral part of the Regulations during the development of the first wide-bodied airplanes.

While the new standards of safety permeate all aspects of airplane design they probably had the most visible effect on the design of the flight control systems. The effect on these systems was even more emphasized by the fact that electronics had developed to the point where automatic landing became a practical reality and all-weather landing capability requirements added further depth to flight control system integrity requirements. Application of statistical failure probability theory to determine flight control system integrity was very much in its infancy during the initial development of these aircraft and therefore the FAA rules were expressed in more general language. However the rule was very specific with respect to the requirement to retain airplane controllability after any single mechanical failure (continued safe flight and landing) and no exceptions hereon were allowed however improbable the failure. The rules with respect to system jams were more subjective and allowed the designer to prove jam alleviation capability or extreme improbability. A similar rule was established for multiple failures, where again retaining control capability or extreme improbability of the failure combination had to be proved. At that time no generally accepted definition of "extreme improbability" was established and the developer had to use conservative judgement in applying the rules. Some parenthetical examples shown in the "Special Conditions" were used as guide lines (e.g. any single mechanical failure in combination with a hydraulic or electrical failure). Later during the development the interpretation of these rules became more specific, particularly with respect to correlation between failures, secondary failure effects and also consideration of higher probability failures which could make consideration of different failure combinations necessary. Furthermore, ways of considering dormant or passive failures became better defined by gradual acceptance of mathematical probability methods.

Where these regulations set the basic safety requirements, actual design integrity requirements were obviously determined by the severity of the potential failure effect and this in turn depended on the overall airplane control configuration. Some features inherent in the basic airplane concept had a profound effect on control system integrity requirements. The most important ones were the size of the airplane, the flying tail concept and the requirement for CAT III automatic landing capability.

- The size made direct manual control impossible even with the most advanced aerodynamic control techniques and even when considered for emergency control only. This made fully powered systems mandatory and for the first time dictated availability of power under all failure mode conditions including all engines out.
- The flying tail concept, by providing a single basic pitch control surface (as compared with conventional independent control of elevators and horizontal stabilizer) mandates a driving system of extreme reliability.
- All weather automatic landing capability makes it necessary that in the final stages of the approach and landing the overall primary flight control system have fail-operative capability with a minimum of pilot attention or need for corrective action.

#### System Configuration and Implementation

Figure 2 shows the general control surfaces arrangement. In designing the system configuration a conscious effort was made to achieve inherent system integrity with a minimum of reliance on additions which themselves introduce new failure modes. Specifically in the pitch control system this has led to some unconventional design characteristics.

**PITCH CONTROL SYSTEM.** The "flying tail" concept for pitch control was selected because of its operational and aerodynamic control advantages. It requires summing of short term pilot and long term trim inputs at the input level providing one integrated input command to the single pitch control surface. It, thereby, prevents the problem of the pilot column and the trim system generating contradictory commands at the aerodynamic control level and inherently solves the potentially dangerous "pitch upset" and "mistrim at take-off" problems associated with the conventional separate elevator and horizontal stabilizer pitch control systems. However, this same absence of multiple control surfaces, demands a high degree of integrity of the pitch control system, particularly when required to meet the contemporary safety requirements. Nevertheless, after many trade-off studies it became clear that the direct approach of providing a single stabilizer control system with a very high degree of system integrity would offer the best and safest solution. This required a system capable of absorbing all conceivable single component failures and a variety of multiple failures, without becoming inoperative and without introducing inadvertent hazardous pitch commands. The following paragraphs describe the system configuration with a short explanation of the underlying rationale. A schematic diagram of the pitch control system is shown in Figure 3.

To make full use of the potential of the flying tail control concept the system had to be configured to give the pilot full stabilizer control through the column independent of trim input. This requires that the column input can override or negate any trim input and still provide full control. An obvious way would be to provide 100% parallel trim as is often done in roll and yaw control systems. This means that there would be a fixed relationship between column and stabilizer and that the trim system would only control the zero force datum point of the artificial feel system. The pilot could then control the stabilizer to any position by applying sufficient force to override the feel system. However, some other constraints on the system make this unacceptable. The integrated control system, including the trim and feel subsystems had to comply with the following objectives:

- Provide sufficient control capability through the column to enable the pilot to take off, fly or land the airplane with maximum mistrim.
- Allow a maximum variation in zero force column position of 3.8 inches (determined by flight simulation tests).
- Provide a minimum column to surface position gain of 1 inch per g.
- Provide artificial feel forces to provide a minimum of 33 lbs/g in cruise and 50 lbs/g with flaps down.
- Keep the variation in stick force per g as a function of c.g. and flight condition to a minimum.

Complying with all these objectives resulted in a non-linear relationship between column and surface (J-curve) with a 50-50 mixture of parallel and series trim and an artificial feel force controlled as function of trim angle and Mach number. Figure 4 shows the non-linear relationship curve and the horizontal shift of that curve as a result of the series trim input. The "Trim Line" shown in the figure depicts the column to surface relationship for zero column force. The relationship between column and surface for each trim position can be represented by a line through the trim point horizontally parallel to the B and A lines. These lines show the relationship for minimum ( $0^\circ \delta H$ ) and maximum ( $10^\circ \delta H$ ) stabilizer trim angles respectively. This non-linear relationship and the way the trim system interfaces with the main pitch control system has the added advantage that no special measures were required to guard against a trim runaway. One fixed maximum trim motor speed provides automatically the maximum required and maximum allowable trim rate for all flight conditions.

In contrast to all other commercial airplanes the L-1011 trim system provides a rate/position control system for the flight crew via trim knobs on the columns. Since the actual rotation of this knob generates the input command no trim switches are involved and a trim runaway can not be caused by a faulty trim switch. Direct mechanical trim control is available by means of the conventional trim wheels on the center console.

After the non-linearizer/trim interface had provided a superior pitch control configuration from the point of view of basic control qualities, the task remained to implement this configuration in order to give it the necessary operational integrity. The following aspects had to be considered:

- Provide sufficient hydraulic and mechanical redundancy to remain fully operative after multiple hydraulic failures (loss of 3 systems), mechanical failures such as link breakage, cable failure, loss of center support of cranks or combinations thereof.
- Retain operational capability in case of any conceivable system jam. This includes the surface/power servo system loops as well as the input system.

It is not intended to mention here all the design details that were incorporated to enhance the total system integrity. Only special features will be discussed and in the next paragraphs the more important component details will be touched on in sufficient depth to give an impression of the need for detail design considerations necessary to meet the safety standards.



**Stabilizer Power Servo System.** A quadruple power servo system was considered essential. Apart from providing a high degree of power redundancy in case of loss of multiple servo channels or their power sources, e.g. as the result of an engine explosion, collision or criminally-induced structural damage, it also offered the potential of a reliable jam-breaking capability in case of a linear hydraulic actuator jam. Each of the four servo channels is powered by a separate and independent hydraulic power source and each servo loop by itself is able to control the airplane. The system is split up in two dual units (each consisting of two linear actuators and one servo valve unit with feedbacks) located approximately eight feet apart, which drive in unison the center section of the horizontal stabilizer. The physical separation was considered necessary to provide protection against localized destructive forces (explosion or fire) but posed the problem of synchronizing the inputs to the servo units. This problem was somewhat facilitated by holding the stiffness of the stabilizer section to a maximum of about 30,000 lbs/in, which allowed some mismatch in the servo position outputs without building up excessive force differences between them. Each servo unit contains the necessary valving and accessories of two servo loops. The servo units are designed in a way that the rupture of a pressure vessel cannot result in the loss of the fluid of more than one hydraulic system, each of which is contained in a separate manifold (this feature is applied to all other hydraulically redundant components). The main control valve is a tandem valve controlling two servo loops, whose actuators are installed side by side. The output path of each actuator incorporates a very accurately manufactured shear pin at the lower connection which is capable of supporting safely the maximum output of a single actuator, but which can be reliably sheared off by the force of three actuators when one actuator is mechanically or hydraulically locked. The actuators are installed in pairs near their servo valve units. Both actuators of a pair are mechanically connected to each other to force a disconnected actuator to retain its orientation in the airplane to prevent major secondary damage. The interconnecting linkage is strong enough to support the load of an actuator if the upper attachment is lost (e.g. loss of the pin). The two main bearings of the stabilizer have been designed to provide two additional concentric rotating bearings to allow stabilizer motion without damage to the bearing structure in case the main bearing or even the main and secondary bearings freeze mechanically. The stabilizer mechanism is further shielded from large objects which could lodge between the structure and the stabilizer, while smaller parts will be easily crushed or sheared by the tremendous torque output capability of the servo system (approx 70,000 ft. lbs) without inflicting major damage to tail or stabilizer structure.

The system is protected against a servo valve jam (which affects both servo channels at one side) by a self monitoring hydromechanical system. A force on the valve of over 100 lbs will activate this system by displacing a spool inside the main control valve spool. This triggers a chain of hydromechanical events which closes the main pressure shut-off valve, places the actuator in a bypass position, operates a lock valve which in turn latches this condition (to prevent reengagement when the force on the valve disappears) and through a mechanical interconnect disables the shut off mechanism of the two remaining operative servo channels at the other side. This last feature prevents the automatic system or the pilot from ever inadvertently deactivating the total system. The relative spool displacement can be caused by a force on the input arm relative to the (dual) feedback arms but the same thing happens when the two feedback arms are desynchronized as the result of a failure (breaking or bending of an arm, etc). Since this monitoring system is normally always dormant, means are provided to introduce an artificial valve stop and verify proper system operation. This stop can not inhibit valve motion when the servo is pressurized and a special procedure is required to make it operative. This procedure can not be applied in flight but as an additional protection the stop is introduced well beyond the normal valve control stroke in the overtravel range. By a periodic check proper operation of the valve jam protection mechanism as well as the protective servo interlock mechanism are verified.

A special hydromechanical lead-lag network is incorporated in each servo loop to improve the stability of the servo system, which drives the heavy horizontal stabilizer (with very low frequency structural modes) by, for this mass, relatively small actuators. However, properly phased structural feedback effects and valve damping make the system inherently stable even in the worst failure modes and therefore special redundancy and/or check out mechanisms for these networks were unnecessary.

**Stabilizer Servo Input System.** The stabilizer servo system is driven from conventional dual column and dual cable systems. The cable systems run at opposite sides of the fuselage and after penetration of the aft pressure bulkhead and attachment to the feel system, are coupled together by an aft-synchronizing mechanism. At both sides of this coupling mechanism the series trim input is summed with the pilot input to drive the input arms of both servo modules via separate non-linearizers. Because of the importance of keeping the four channels of the servo system synchronized, the total mechanism connecting the two valve modules is very rigid and mechanically dual redundant. This includes the series trim system mechanism itself which must provide the reaction force in the summing link.

Providing mechanical redundancy to protect against mechanical failure is relatively simple but it is much more involved to keep a single system operative in case of a mechanical jam. Since protection against inadvertent automatic disengagement of a self correcting input jam correction system became very involved and since in this case the requirements could be met with a pilot warning, automatic instruction, and a manual correction system, this last method was selected. The warning system derives its intelligence (to determine location of jam) from three two-way bungees, one in the front of each cable path and one in the servo synchronizing link. These are actuated by a higher than normal pilot force. In most cases of a jam in the input system, control is still possible through the series trim component of the trim system. Only a jam in the coupling mechanism between the power servos requires pilot action before significant pitch control inputs can be made. In that case initial corrective action by the pilot consists of deactivating two stabilizer servo channels as instructed by the detection and warning system. This action simultaneously and automatically separates the two servo inputs by means of a decoupler in the servo synchronizing linkage allowing immediate control from either column, but requiring overriding one of the front bungees. However a second instruction appears at that point to open a coupler between the two columns. After decoupling, normal control is restored with normal trim but with one half of the feel force and from one column only.

The artificial feel forces are generated by multiple mechanical leaf springs. There are two units, each one directly connected to one of the cable systems upstream of the servo synchronizing mechanism and each one incorporating three leaf springs. The mechanical advantage from column to spring is determined by a variable gain mechanism. The gain of the mechanism is mechanically controlled by the stabilizer series trim input and in addition is modified as function of airplane Mach number by means of electric motors. Figure 5 shows the force fan curve. The mechanism is designed so that in case of complete failure of either the series trim system or the Mach feel drive system, the feel forces for landing will always arrive at approximately the nominal value, providing inherent fail operational system design. The Mach feel system is dual redundant with only a single system needed for operation. The outputs of both motors are summed in a differential gearing after each one has been rendered irreversible through a worm gear for isolation. The feel and trim systems are integrated into each of two units, coupled across the ship by drive shafts. However, the series trim drive is completely integrated into one of the units and will always provide the same input to both servo units in contrast to the parallel trim which can operate differently on one feel unit if the two "trim and feel" units become separated by failure of the connecting drive shaft.

**Pitch Autopilot Servo.** Until the advent of automatic landing systems the only reliability requirement of autopilot servos was that it be fail-safe in all conditions. This was mainly accomplished by restricting their force output authority to a safe low value and by providing the capability for pilot override at any time. This was based on the assumption that the flight crew would always be in a position to take over the piloting of the airplane, that is, the autopilot provided only a convenience function. With the advent of automatic landing the picture changed. Not only did the autopilot need a large authority but in addition its function and the completion of its task became essential under some circumstances. Therefore a new concept of autopilot servo had to be developed. Based on previous experience, Lockheed elected to stay with electro-hydraulic servos driving the power control input systems in a parallel manner. Because of the peculiar configuration of the horizontal stabilizer power servo system, with the physically separated valve modules, the conventional way



of integrating a modulating piston servo within the power servo unit was not considered applicable. Without complex electronic synchronization the potential differences in input commands could lead to very high fatigue loads between the two sets of power servo actuators and could adversely affect the achievable increment control. For this reason the mechanical autopilot commands resulting from the multiple channel autopilot servo system were integrated into a single mechanical command in a separate autopilot servo unit. This mechanical command output is introduced in the main power control system in the servo synchronizing mechanism thereby assuring a synchronized command to the power servos.

The pitch autopilot servo system (refer to Figure 6) consists of two electrohydraulic servo channels which are coupled through force limited couplers. To allow synchronization errors in the individual channels and mod piston positions the coupler of the channel designated as "secondary" allows a limited motion in the coupler by applying half the force of the basic coupler. The result of this feature is that when both channels are operating the "coupled" output will always track the primary mod piston exactly while allowing a certain mistracking of the secondary mod piston. This prevents upsetting the electronic in-line monitoring systems and allows perfect increment control. When the primary system fails, or the autopilot operates on the secondary channel only, the limited force/limited stroke coupler of that channel has sufficient coupling force to drive the output as a fixed link. The coupled mod pistons output is connected to the mechanical system output via an "engage and override mechanism." This couples the autopilot system with the primary system with an accurately controlled and variable limited transfer force. In those modes where the autopilot provides the conventional convenience function, the authority is still limited to a safe value (approximately 21 lbs override force at the column) but in the approach and landing mode, where maneuvering capability is required, the authority is increased to approximately 44 lbs. The engage mechanism is designed so that the force is practically independent of the number of channels in operation. The coupling force is hydraulically generated and controlled by two-value pressure controllers. The setting of these controllers is again hydraulically controlled. The pressure controllers are backed up by two-value relief valves, the settings of which are switched simultaneously with the controllers. Pressure sensors provide various signals to the monitors and the pilot to indicate the status of the system. Multiple solenoid pressure control of mod pistons and couplers plus the series action of mod piston, mod piston coupler and engage coupler release capabilities provide a virtually jam proof system, and mechanical, hydraulic and electrical redundancies provide the fail-operational characteristics.

**DIRECT LIFT CONTROL SYSTEM (DLC).** During the early stages of the development of the automatic landing system it became clear that a Direct Lift Control system, implemented by control of some of the wing spoilers, would be advantageous in reducing the touchdown point dispersion in turbulent air conditions. Therefore the primary flight control system specification included such a system from the beginning.

Since wing spoilers are used for air brakes, ground lift spoilers, roll augmentation, and direct lift control all these functions had to be integrated in the various systems design. In Figure 7 a schematic is shown of the mechanical interface of these systems. Because of aerodynamic characteristics the combination of the four most inboard spoilers on each wing were best suited for lift control with minimum pitch effect. A separate DLC servo system has been developed which normally acts as a pilot assist servo for the speed brake system, but also accepts the electrical commands for automatic ground spoiler deployment and direct lift control. The servo system combines two dual redundant hydromechanical servo channels with two electrohydraulic channels, integrated in such a way that a hydromechanical and an electrohydraulic loop share the same actuator. The hydromechanical servos provide the speed brake pilot assist control. The tandem valves are normally locked in neutral during DLC operation. However a pilot force of about 24 lbs on the speed brake lever overrides that valve locking mechanism. The flow capability of the mechanical valve is twice as large as that of the electrohydraulic valve so that the pilot override action can wash out the effect of the electrical servo loops and determine the servo output without the need for immediate disengagement of the electrical loop. A switch on the speed brake lever enables the pilot to disable the DLC servo system completely. In that case the pilot assist as well as electrical command modes are deactivated, but direct mechanical "flow through" command to the spoiler power servos is still possible.

During operation in the DLC mode, a spoiler deflection authority of plus or minus 8 degrees around an 8 degrees spoiler-up bias is required and the airplane is protected against larger inputs by DLC servo actuator stroke limiters. These limiters are hydraulic pistons activated during DLC operation. They are deactivated for automatic ground spoiler deployment by a positive signal generated after the necessary interlock requirements have been satisfied. Otherwise only a pilot override action can remove the stroke limiter.

Figure 8 shows the interlock diagram for a single channel of the DLC. This circuit will automatically activate the direct lift control system in the approach (throttles retarded, flaps down etc.) and also controls the automatic deployment of the spoilers after touchdown. In addition it provides the multiple safeguards against inadvertent spoiler deployment in flight, prevents full deployment of the spoilers until both wheels are on the ground (limiter control) and deactivates DLC automatically when required by flight condition (stall warning or go-around). Furthermore the in-line monitors deactivate one (or both) channel(s) when errors in the position modulation commands are detected. The flight crew has the capability to permanently deactivate the system(s) by means of control switches on the overhead Flight Control Electronics Systems panel. These switches are normally in the "on" position during the whole flight.

To minimize the effect of a speedbrake input cable failure, which, because of the stored energy, may cause a spoiler-extend command, the cable system is separated in three individual cable loops in series. This reduces the energy released in case of a cable break, by a factor of three and prevents overshoot of the servo input beyond the static position entirely. All valves and spoiler servo inputs are spring-biased to cause spoiler retraction in case of physical disengagement from the system and spoiler servos are controlled and disengaged in pairs (one left, one right) to minimize unwanted roll effects in case of failures.

**ROLL AND YAW CONTROL SYSTEMS.** The roll and yaw control systems, are more conventional in configuration and therefore are only briefly touched upon. Figure 7 shows the configuration layout of the roll control system. The left inboard aileron servo system, with three individually powered servo channels, serves normally as pilot-assist servo for the complete lateral control system, driving the inputs to the left outboard and right inboard aileron servos and the left hand mixer mechanism, which controls the activation of the inputs to the number 4 left and right spoiler servos. Moreover it drives the right cable system, which is normally decoupled from the control wheels by a lost motion device. The right inboard aileron servo in turn drives the inputs to the right outboard aileron servo and the right hand mixer mechanism controlling the number 2 and 3 spoiler servos input activation. Left and right selector mechanisms shift the inputs of the number 5 and 6 spoiler servos between the outputs of the inboard aileron servos and those of the number 4 spoilers. The table in Figure 7 shows the schedule of activation of all spoilers. The number 1 spoilers are never used for roll augmentation and are therefore driven directly by the DLC servo. However, since they are only used for direct lift control and ground spoilers and not for high speed air brakes they are electrically deactivated in all flight conditions unless landing flaps are selected. The aileron and spoiler servo systems are powered by the hydraulic power sources in a way which requires the presence of only three hydraulic systems in any one wing and still does not introduce a significant rolling moment when hydraulic power of one or more systems is suddenly lost. This again protects against potential loss of all hydraulic power in case of local structural damage.

The dual channel electrohydraulic roll autopilot servo system is integrated with the left inboard aileron power servo system which drives the total lateral control system. Although mechanically implemented quite differently from the pitch autopilot servo system, it is

also configured on the principle of force summing of the outputs of both channels, with an allowance for mistracking of the "secondary" channel actuator by a limited force/limited stroke override. The system incorporates an hydromechanical device which will automatically deactivate a servo channel in case of a runaway, hardover or output jam.

The mechanical input system incorporates a torque limiter upstream of each cable system which allows an input into one half of the system when the other half is jammed. Each limiter incorporates a sensor which when activated signals the general location of the problem by lighting instructions and aileron and spoiler servo control switches on the overhead control panel. The flight crew can then separate the two control wheels and maintain control without the need for overriding the torque limiter. This action automatically closes up the lost motion mechanism. The override bungee between the left inboard aileron servo output and the right inboard aileron servo input acts as secondary feel spring when the left control channel is deactivated.

The rudder power servo control system, also powered by three independently powered servo loops, is very similar to the left inboard aileron servo system. However the two electrohydraulic servos are tied in to the main system in a way to provide a series input for the yaw stability augmentation system and are switched to a parallel mode (autopilot) only during the last part of the approach for automatic runway alignment and rollout when full rudder control may be needed.

### AUTOMATIC FLIGHT CONTROL

The Avionic Flight Control System (AFCS) of the TriStar provides for manual and automatic control of flight guidance functions and related monitoring and warning functions throughout the total flight envelope, from takeoff through landing roll-out. This is accomplished with an integrated set of subsystems which combine to provide a reliable system that increases aircraft safety by reducing crew work loads and by providing accurate control to desired flight tracks. The AFCS hardware, which was developed by a Collins Radio - Lear Siegler team, has achieved the goal of providing the operational performance and reliability required for a Category III automatic landing system.

The L-1011 AFCS consists of four integrated subsystems. These consist of:

- Stability Augmentation System (SAS)
- Autopilot/Flight Director System (APFDS)
- Speed Control System (SCS)
- Flight Control Electronic System (FCES)

The yaw stability augmentation system provides yaw damping and turn coordination throughout the entire flight regime. In addition, the yaw SAS computers contain the Autoland<sup>®</sup> functions of runway alignment and rollout. The autopilot/flight director system provides guidance (flight director) and automatic flight control modes for the entire flight profile, including Category IIIa automatic landing. The speed control system provides automatic throttle computations as well as takeoff and go-around capabilities. The primary flight control electronic system comprises several subsystems, including pitch trim (manual, automatic, Mach), Mach feel compensation, altitude alert, stall warning, direct lift control, automatic ground speed brakes, and primary flight control monitoring.

The components which comprise the AFCS are listed by subsystem in Table 1. For total system function, these components interface with other airplane elements such as air data sensors, attitude references, radio navigation and altimetry systems, electrohydraulic and electrical flight control servos and flight instruments. Figure 9A is a photograph of the 48 separate AFCS items. The nine computers are all 3/4 ATR long units providing standardization of many mechanical as well as electronic components. The computers comprise 68% of the 261 pound total for all the items shown.

Functional element circuit implementation and modular packaging are features of the computers. The hardware, designed during the early phases of L-1011 development, uses D.C. analog computations and digital logic. Figure 9B is a photograph of the pitch computer chassis, illustrating the construction common to all AFCS computers. A multi-layer printed circuit sideboard provides the interconnect for the plug-in printed circuit cards. The sideboard wiring is physically separated for each of the redundant channels to ensure maximum integrity. The separation is maintained within the computer connectors and throughout the aircraft wiring as well as on the circuit cards.

Figure 9C is a photograph of one of the printed circuit cards. A clear area is provided between the redundant elements although in some cases monitor circuits are mounted in the center of the card.

### STABILITY AUGMENTATION SYSTEM (SAS)

#### System Mechanization

Figure 10 depicts the cruise configuration of the SAS. Each of the two yaw computers contains two computations that output identical servo commands to an in-line monitored electrohydraulic servo. Four aileron position transducers and three rate gyros service the four computations of the total system. The rate gyros provided for Dutch roll damping inputs and the aileron transducers provide for turn coordination.

Figure 11 shows a SAS cruise computation. It is seen that the gains are scheduled with flap position and the gyro path has the usual low frequency washout filter plus a high frequency cut-off. The aileron input path has a limited washout to remove aileron trim effects and a scheduled dead zone such that turn coordination only comes through sufficiently large aileron inputs. The passed signal is subject to gain changing to match the gyro path and to low pass filtering. The voter output to rudder surface response can be approximated by a two Hz second order servo for small amplitudes. However, the primary control surface servo is severely hinge limited in cruise flight.

It is noted that in Figure 11 the output of a computation comprises one input to a voter. As one would expect, there are two voters per yaw computer with two voter outputs required to drive one SAS electrohydraulic servo as depicted in Figure 12. The two voter outputs provide for driving the electrohydraulic valve (EHV) coils in a push-pull arrangement with two sets of redundant monitors acting to shut off the servo loop hydraulics if a fault is detected. One set (CS1) monitors the unbalance between the two halves of the servo drive and the other set (CS2) monitors input command versus output position.

Figure 13 shows the voter input crossfeeding for the complete dual-dual system. There are redundant monitors in front of the voters which control the signal configuration of the voter inputs as shown in Figure 14. This figure illustrates the concept whereby the monitors control switching logic that substitutes signal ground or an alternate computation for a faulted channel.

In addition to servo and computation monitors, there are rate gyro monitors and electrical power monitors. The latter operate into the servo engage logic while the former monitors operate into the voter switching logic.

It is seen from the foregoing that the voters are important to this dual-dual arrangement in performing two very significant functions.

- The voters equalize the four computation output signals which are then used by the in-line monitored servos.
- The voters pass a good signal while a faulty computation is being detected by the pre-voter monitors.

Both these functions are essential to minimizing nuisance monitor trips with acceptable monitor detection levels.

#### Design Objectives and Performance

The function of the cruise mode of the SAS is, of course, to provide improved Dutch roll damping for enhancement of passenger comfort and handling qualities and for reduction of loads on the vertical tail. This reduction of fin tail loading, in continuous turbulence, was reflected in the definition of limit design loads.

Early in the development of the L-1011, the effectiveness of the SAS was investigated to determine performance and reliability objectives for the SAS from a loads viewpoint. It appeared that a minimum damping ratio of 0.3 and a timewise availability of 97% were modest design objectives that would yield significant load reductions. It was subsequently found, however, that higher damping ratios could be achieved over most of the climb, cruise and descent flight regimes as seen from the data given in Table 2. Only at low speeds, where effects on fin loads are not critical, are the damping ratios less than 0.3.

It also became evident that a 97% availability requirement was a very conservative estimate of system reliability. The single channel failure rate was calculated to be about  $10^{-3}$  per hour and to preclude the possibility that an airplane might be flown without SAS for a protracted period, it is required that at least one of the two channels be operative for dispatch. Recognizing that for most flights both channels of SAS are operative, even 99.9% timewise availability would appear to be conservative.

A complete discussion of the effect of SAS availability on loads is given in Reference (1) from which Figure 15 is taken. This figure illustrates the definition of design loading for vertical tail shear with 0, 97 and 100% SAS availabilities. It is based on a mission analysis criterion whereby the frequency of exceedance of a load quantity is calculated for operations over specified design flight profiles. The turbulence environment as statistically described for each segment of a profile is applied to the airplane/load transfer function to derive exceedance curves (with or without SAS operating) for each segment. The segment exceedances are summed over the total of all profiles to determine a load vs. frequency-of-exceedance curve for the mission.

It can be seen from Figure 15 that the major reduction ( $H/F = 0.70$ ) is realized by having at least 97% availability and further reduction comes less readily with 100% availability realizing a ratio of  $G/F = 0.65$ . These results are for a fully linear system and saturation effects reduce the benefits somewhat. In summary, however, with 97% availability the net reduction in fin loading is better than 25% compared to what it would be if no SAS were available. In actual practice the SAS availability has proven to be much better than 99.99% leading to the conclusion that for all practical purposes the load alleviation is in accordance with the 100% expectation.

#### AUTOPILOT/FLIGHT DIRECTOR SYSTEM (APFDS)

The APFDS consists essentially of two integrated autopilot/flight director channels whose operation is controlled from a single set of control panels mounted in the glareshield. The hardware includes two identical pitch and roll computers each of which contains a single "cruise" computation and dual automatic landing computations. The computations are common for both autopilot and flight director control; however, some separation is retained where required to provide performance unique to the autopilot or flight director function.

The APFDS provides the usual pitch and roll modes:

Roll and Pitch Attitude Hold with Control Wheel Steering (CWS) and Turbulence Configuration Control

Altitude Select and Hold

Vertical Speed Select and Hold

Airspeed Hold

Mach Hold

Heading Select and Hold

VOR and Area Navigation (Lateral and Vertical)

Localizer Capture and Track

In addition, there are the common axis modes of:

Approach

Approach/Land (Automatic Landing)

Go-Around

Takeoff

The pitch commands for Go-Around and Takeoff are derived in the Speed Control Computer (SCS) with Takeoff being a flight director mode only. The Approach/Land mode is used for automatic landing and is the only fail-operative autopilot mode. The remaining modes are the "cruise" modes; their mechanization is loosely referred to as the cruise autopilot.



## CRUISE AUTOPILOT

### System Mechanization

Figure 16 illustrates the redundancy and monitoring characteristics of the APFDS cruise configuration. Servo monitoring and pre-voter monitoring and switching are not shown in Figure 16, these features are, however, similar to those shown in Figures 12 and 14.

During cruise operation, only one autopilot servo can be engaged; however, both APFDS channels are available for independent flight director control. As shown in Figure 16, if channel "A" autopilot is engaged, the "A" cruise computation is processed to a completely redundant inner loop comprised of quadruplex command limits and voter/monitor functions. The pitch axis limit is on attitude rate command while the roll limit is on aileron command. The voter outputs provide redundant signals to the in-line monitored servo loop. Therefore, failures of the single cruise computation are command limited for fail-safe performance. Fail passive performance is provided for failures of the redundant inner loop or servo system. Over and above the protection provided by the command limiters there are back-up fail-safe provisions to assure that hazardous failures are indeed extremely improbable. The back-up autopilot safety feature for the pitch axis consists of an autopilot servo authority limit. The servo output coupler can apply only a limited force to deflect the controls, the force being resisted by the feel spring. Under the most adverse conditions (high Mach, high q) the transient responses to any AFCS failures are thereby limited to less than 1 g (delta). The back-up provision for the roll axis consists of automatic autopilot disconnect in the event that the total aileron angle exceeds 28 degrees (70%). The detectors in this circuitry are dual switches (for each autopilot) either of which can effect a disconnect through independent circuits. It is necessary to override these limiters, however, during automatic landing, for example, and in CWS with a pilot wheel input applied. These pitch and roll back-up provisions and the inherent rate limiting of the control system servos adequately limit the airplane response to hardovers, slowovers and oscillatory failures.

The basic mode of the autopilot is attitude/heading hold with control wheel steering (CWS). In order to alter his reference, the pilot need only apply control wheel force in the desired axis and maneuver the aircraft to the new reference. No manual mode select or knob twist is necessary. It is noted that while pilot force is applied, the command limits are increased so as not to restrict the pilot's maneuvering capability. Redundant fail-safe detector circuits provide for adjustment of the limits.

The engage features of the APFDS provide increased flexibility by allowing a coupled command mode in one axis and CWS in the other. This expands the operational capabilities and eases pilot work loads in maneuvers such as holding patterns, where, for example, altitude hold could be used in pitch with CWS retained in roll. Since the system provides a separate CWS engage position, flight director command modes may be displayed while using the CWS features. A softened CWS mode is provided by the turbulence mode. This enables rough weather penetration with attitude hold features at reduced control activity levels.

The altitude capture system has been integrated with the altitude alert system. This allows a pilot to set a clearance altitude, automatically capture and hold that altitude, and arm the altitude alert function with a single pilot action.

The localizer and approach modes provide Category I capability for use on beams which are not suitable for automatic landing and for non-precision approaches. The approach/land mode provides Category II approach capability and the fully automatic landing and rollout functions. The takeoff mode provides flight director guidance for programmed climbout capability while the go-around mode provides both automatic and flight director missed approach capability.

Figure 17 shows the AFCS controls and displays. A single surface position indicator is provided to indicate the position of trim and all of the primary control surfaces. A single set of autopilot/flight director and speed control panels are located on the glare shield. These consist of five panel modules, including Thrust (auto-throttle), Heading Pitch Mode, APFDS Engage, Navigation Mode, and Altitude Select panels. In addition, two AFCS mode indicators and two AFCS warning indicators are provided, one on each pilot's panel.

Each autopilot may be engaged via the appropriate solenoid held bathandle in either the CWS or command configuration. Flight director indications are controlled by independent selector switches also located on the APFDS engage panel. The autopilot may be disconnected at any time by means of disconnect switches located on each control wheel or by use of the bathandles. The autothrottle system is engaged by its own solenoid held bathandle and may be disengaged via the bathandles or by disconnect switches on the throttle levers.

Since the AFCS consists of two integrated autopilot/flight director systems, each flight director necessarily displays the same mode as its respective autopilot. Combined with the requirement that the two autopilot/flight director systems remain mode synchronized, an effective pilot interface is achieved by the use of a common mode controller for both APFDS systems. Redundant contacts on each mode select button maintain the integrity and operational availability advantages of a dual autopilot installation. Mode selection is annunciated by an illuminated mode select button and is confirmed by the two APFDS mode annunciators.

### Design Objectives and Performance

The safety objective for the cruise autopilot was to achieve AFCS fail-safe performance such that the backup failure protective means would be needed less than once per  $10^7$  hours. This would assure that potentially hazardous events would be extremely improbable. This  $10^{-7}$  per hour system failure rate was readily achieved because of the basic redundancies which were obviously needed to meet the Category III requirements.

The significant failures are downstream of the command limiters and voters in the servo area and in the disconnect circuits. Figure 18 shows the disconnect circuit redundancy for one autopilot. It is seen that there are two valid/logic paths either of which can cause a disconnect. Path A holds the bathandle solenoids engaged and path B engages the servos. There are four ways to deenergize the servo solenoids - removing servo solenoid grounds, removing bathandle solenoid grounds, actuating control wheel disconnect switches (captain's and first officer's) or pushing the bathandle to the "off" position.

During certification testing, demonstrations of the three basic failure protection methods were performed. Hardovers were applied upstream of the command limiters to show the responses at the limit values. Faults were applied in the servo circuits to demonstrate their fail-passive nature. Hardovers and slowovers were applied with monitors by-passed to verify the fail-safe characteristics of the back-up protective means. Oscillatory failures at frequencies above which the various failure protection methods would not work were cleared by showing that the maximum rate capabilities of the various servos could not provide significant amplitudes.

## AUTOMATIC LANDING SYSTEM (ALS)

### System Mechanization

The principal elements of the ALS are the APFDS and SAS and their respective sensors in the configurations established below 1500 feet (radio altitude) with the Approach/Land (A/L) mode selected. As seen from Table 3, the system in total definition includes



much more than these units but these have the most effect on system reliability and availability. Reliability is used here in the sense of the system capability to complete a landing. It relates directly to safety, particularly in low weather minima operations. It was, of course, the Category III requirement that dictated the extent of ALS redundancy which is depicted in some generality in Figure 19 for the pitch and roll control axes. Each of these axes uses three accelerometers (normal or lateral) and three attitude inputs. Pitch computations use only derived pitch rate; roll uses both attitude and derived rate signals. The primary Autoland <sup>®</sup> Sensor signals are glideslope (G/S) error and radio altitude for pitch and localizer (LOC) error for roll. Only two each of the Autoland <sup>®</sup> Sensors are used but each has dual outputs with high integrity self-monitoring. For example, the probability of the two signals from one G/S receiver being faulted at a critical time without warning signals to the pitch computer is less than  $10^{-9}$ .

The same theme of APFDS redundancy is carried over into the SAS in the A/L mode as seen in Figure 20. Here, the exception is that only two compass systems are utilized which do not have the integrity of an Autoland Sensor. The redundancy requirement, however, is not as great for yaw control as it is for pitch and roll. Automatic landings with no automatic yaw control have been demonstrated without any significant effect except that the pilot had to control the alignment and rollout. The compass inputs are actually compared and averaged in the SAS computers and used to define a reference heading error which is memorized. The compass signals are switched out at 150 feet and integrated rate gyro data is used from there to touchdown. (The radio altitude signals used to control the transition are omitted from Figure 20.) Below 150 feet, a maneuver is performed whereby the aircraft fuselage is aligned with the runway and a limited wing down is held against crosswind. Up to eight degrees of crab can be removed in this manner. The basic inputs for this control are heading error and localizer error and the latter continues to control the airplane in rollout through rudder and nose wheel steering. This rollout mode which is effective to low ground speed is a safeguard against patchy or swirling patterns that might temporarily cause loss of visual reference. It is also necessary for extending the system capability to Category IIIB.

As would be expected, the fail-operative pitch, roll and yaw (below 150 feet) mechanizations closely follow that as depicted in Figures 12, 13, and 14 for the cruise yaw control. Four computation channels for each axis are needed for the fail-operative condition and two or three for the fail-passive condition. The latter configuration is acceptable for Category II operations while the former is required down to the alert height for Category IIIa. There are minor differences in each servo control and monitoring mechanizations, but the basic concepts of Figure 12 are applied. For Category III, of course, it requires two servos per axis while one is acceptable for Category II.

*Although not considered as necessary to automatic landing as pitch, roll, and yaw control, the functions provided by the SCS are important to Category III as automatic throttle is required as is automatic go-around in some cases.*

There are two throttle control modes — airspeed hold with pilot selectable reference and stall margin angle-of-attack hold with the latter mode acting as a floor on the former. Airspeed hold is used normally until landing flap is selected at which time the stall margin mode is forced if the autothrottle system (ATS) is engaged. The angle-of-attack reference corresponds to a speed of from 3 to 10 knots above reference speed (depending on airplane center of gravity) and is not adjustable except automatically when spoilers are operated (for DLC) and when down and/or tail gusts are detected. The "gust sniffer" circuits can add the equivalent of about 4 and 8 knots to the reference speed for 20 seconds (or longer if turbulence levels are continuously detected). When the airplane altitude reduces to 50 feet (radio), a controlled deceleration is programmed until touchdown when the throttles are automatically brought to ground idle and the system automatically disconnected.

For the stall margin mode the computation output commands a throttle rate. With much the same computation scheme the SCS commands a pitch rate when the go-around mode is engaged. In this case the basic reference is an angle-of-attack corresponding to around 1.25 stall speed.

It is noted that a more complete description of the system, that includes the basic control laws, is given in Reference 2. The control laws determine the performance in the presence of disturbances which is fundamental to safety but rather than present them herein, it is elected to introduce three of their features which are a part of the L-1011 ALS to improve performance and safety.

It was pointed out in the discussion on Primary Flight Control that one of the subsystems of the AFCS is Direct Lift Control (DLC) which was included in the basic L-1011 design for the purpose of enhancing the performance and safety of the aircraft during approach and landing. This subsystem is commonly associated with automatic landing and so is described in this section. It is utilized, however, whenever landing flaps are selected without or with the APFDS engaged in any mode.

The basic DLC mechanization scheme is shown in Figure 21. Stabilizer motion relative to the pitch trim position is measured electrically and after shaping is used to position the DLC servo which drives the linkages to the eight DLC spoilers. The two stabilizer motion transducers are each dual; they are, in fact, the same ones which are used for the fail-operative autotrim system. The DLC servo is also dual (each in-line monitored) and the computations are dual-dual although voters are not required as the computations are simple and track adequately. The system response to pitch control is invariant whether pilot or autopilot moves the controls.

*Two means are employed to minimize the effects of ILS signal anomalies.*

- Radio inputs to the computations are held at fixed levels for short periods (1 second for G/S and 5 for LOC) before a disconnect is effected because of an ILS radio flag.
- Large LOC deviations (without a flag) result in clamping the radio inputs for up to 8 seconds before normal tracking is resumed. The level of "large" deviation which causes the clamping is scheduled with radio altitude, closing to  $\pm 20$  microamps near the threshold.

The need to cope with ILS anomalies should be minimal in Category III conditions but these do, however, seem to be prevalent, particularly beam "overflight" disturbances, in the less severe weather conditions.

Another safety-oriented feature of the ALS is the manner in which automatic pitch trim is used. At the time the fully redundant ALS configuration is established (1500 feet), a trim bias is inserted in the fly-up direction. The autopilot has to control (nose down) against this bias so that a subsequent disconnect will result in a definite pitch-up maneuver. This also has more significance in Category I or II situation than in Category III conditions.

#### Design Objectives and Performance

The L-1011 program objective with respect to the ALS was to achieve a timely Category IIIa certification with a system having the potential for Category IIIB. The design objectives were to produce an automatic landing system with superior performance, reliability and availability so that the full safety potential of automatic landing could be realized.

The L-1011 has been in service long enough to have proven that the system performs in accordance with expectation. The L-1011 ALS average performance in the real world environment is as follows:

- Longitudinal touchdown, 95% footprint – 800 feet
- Touchdown sink rate
  - average, 1.9 feet/second
  - 95% upper limit, 3.0 feet/second
- Lateral touchdown, 95% footprint – 42 feet

This data applies to the maximum landing flap case (42 degrees). For the 33 degree landing flap case, the data is the same except that the longitudinal touchdown footprint is increased less than 15%.

The above performance data applies to the Category II/III system which requires DLC. The system can be used without DLC in other than Category II/III conditions; however, the longitudinal dispersion is somewhat greater. Although this could be modified by increasing the sink rate at touchdown, the DLC reliability is so high that there is no benefit to having two different control laws (with and without DLC) one of which would give significantly higher touchdown sink rates. The advantage of DLC is illustrated in Figure 22. This figure shows flight test data of landings with and without DLC made under comparable conditions. The systems were similar except in two respects, the use of DLC as mentioned and the adjustment for nominal sink rate at touchdown. It is seen in Figure 22 that the system without DLC has the large footprint with significantly higher sink rates. As the improved controllability of DLC results from the faster acceleration response time, similar benefits are realized in manual flying though these have not been so easily quantified.

The effectiveness of the computation circuits in suppressing LOC "overflight" disturbances is shown in Figure 23. The figure gives results of simulator landings, several hundred per disturbance duration, made with 15 knot crosswind and 2.5 microamp rms radio noise plus LOC hardovers superimposed at low altitude (below 50 feet). An "unprotected" hardover of this magnitude even for a few seconds and with no wind would have the airplane off the runway unless the pilot took corrective action promptly.

Of course, with protective circuits of this nature hardover magnitudes around the detection level give the greatest airplane response but the probabilities of landing off the runway are acceptably remote without pilot intervention. In passing it can be said that the simulator also showed a small but measurable improvement in landing footprint as a result of the noise filtering characteristic of the detection/hold circuit.

In addition to meeting the landing footprint performance requirements it is necessary to assure adequate threshold clearances. This follows readily, as long as the ALS is functioning and the footprint itself is within acceptable limits as it is for the L-1011. In the event of a low altitude disconnect prior to the threshold, however, it is not obvious that the clearance will be adequate unless some nose-up input is automatically applied at disconnect. That such an input is sufficient can be seen by comparing (on a statistical basis) the sink rates at a wheel height of 50 feet (radio) plus  $\Delta t$  seconds for the normally functioning ALS and for a system having a total disconnect at 50 feet. If the sink rates in the latter case are less than for the normally functioning system, the expected ground clearances following disconnect would be greater than for the normally functioning system. This was evaluated by simulation for two values of  $\Delta t$  with limiting disturbances (26 knots wind, 2.5 microamp rms radio noise). The probabilities of the disconnected system sink rates being greater than the normal system rates were  $5 \times 10^{-4}$  for  $\Delta t = 1$  second and  $3 \times 10^{-3}$  for  $\Delta t = 2$  seconds. These probabilities become rapidly smaller as the disturbance levels decrease so that it can be appreciated that considering the probabilities of disconnect and of encountering limiting disturbance conditions the trim bias is sufficient to assure adequate clearance heights following a disconnect even without pilot intervention. Considering that the pilot attempts to land following the disconnect, the trim bias may tend to cause a longer landing distance than otherwise but this appears to be better than the alternative.

As service experience is proving the acceptability of the ALS fault-free performance, so is it proving out its reliability. Certification to Category IIIa by the manufacturer is only an initial step toward achieving certification for use in revenue operations. Each operator must verify his capability to use the system. Three L-1011 operators have done so and at least one other is proceeding toward that objective.

One of the things an operator must show to achieve ALS certification is that the system reliability in the airline environment is compatible with the failure rates used in the Lockheed certification analysis. One indication of this is a comparison of failure rates achieved with those used in the Lockheed certification analysis. In effect, MTBF tracking limits are defined. Table 4 shows a list of MTBF lower limits and their currently estimated values. The data given in this table are for the significant contributors to the total disconnect probability (below the alert height). If the MTBF's of all the listed units were at the lower limits, the total disconnect probability would be potentially a factor of two higher, still within acceptable limits. These "lower limits" are not absolute limits in view of the fact that the two factor does not put the disconnect probability to an unacceptable level and further, one low MTBF value could be compensated by a high one. To a certain extent the limit is a tracking limit to signal for more detailed examination of a potential trouble area.

In the main, however, the airline results have supported reliability predictions. They are also indicative that with emerging maturity the Category III system availability will achieve the L-1011 objective of better than 95 percent. A certain amount of service experience feedback must be achieved before this level is attained, but it appears at this time to be entirely feasible. That is, given that it is desirable to perform a Category III landing, the landing can be initiated and completed on 95 percent of these occasions. The Category II ALS availability is, of course, of an order better than this.

#### REFERENCES

1. Hoblit, Frederic M.: Effect of Yaw Damper on Lateral Gust Loads in Design of the L-1011 Transport. AGARD Presentation, the Hague, 7-12 October 1973, published in AGARDograph No. 175.
2. Mineck, D.W., Derr, R.E., Lykken, L.O., Hall, J.C.: Avionic Flight Control System for Lockheed L-1011 TriStar. SAE Presentation for Aerospace Control and Guidance Systems Committee Meeting No. 30, 27-29 September 1972, published by Collins Radio Company, Cedar Rapids, Iowa.
3. FAA Advisory Circular AC120-28A, "Criteria for Approval of Category IIIa Landing Weather Minima."
4. CAA Paper 360; "Airworthiness Requirements for Automatic Landing in Restricted Visibility Down to Category III."

TABLE 1. L-1011 AVIONIC FLIGHT CONTROL SYSTEM EQUIPMENT LIST

<u>Stability Augmentation System (SAS)</u>	
2	Yaw Computers
3	Rate Gyros
2	Aileron Position Sensors (dual)
2	Rudder Position Sensors (dual)
<u>Autopilot/Flight Director System (APFDS)</u>	
2	Pitch Computers
2	Roll Computers
2	Pilot's Control Wheels
2	Mode Annunciators
2	Warning Indicators
5	Mode Select Panel Modules
1	Normal Accelerometer Unit (triple)
1	Lateral Accelerometer Unit (triple)
<u>Speed Control System (SCS)</u>	
1	Speed Control Computer
1	Autothrottle Servo
1	Longitudinal Accelerometer Unit (dual)
<u>Flight Control Electronic System (FCES)</u>	
1	FCES Computer
1	Trim Augmentation Computer
2	Angle of Attack Sensors
2	Stick Shakers
1	Surface Position and Pitch Trim Indicator
10	Surface Position Sensors
3	Control Panels

TABLE 2. L-1011 DUTCH ROLL CHARACTERISTICS WITH AND WITHOUT YAW SAS

FLIGHT CONDITIONS (MID CG)							DUTCH ROLL MODE DAMPING RATIO AND DAMPED NATURAL FREQ			
CONFIGURATION	SPEED KEAS	MACH NO.	ALTITUDE KFT	WEIGHT KLBS	FLAPS DEG	GEAR	SAS $\zeta$	ON fd Hz	SAS $\zeta$	OFF fd Hz
Climb	246	.45	10	404	UP	UP	.32	.13	.09	.15
Climb	356	.65	10	308.5	UP	UP	.72	.14	.12	.22
Climb	358	.8	20	400	UP	UP	.56	.15	.07	.21
Cruise	310	.86	33	350	UP	UP	.45	.18	.10	.20
Cruise	260	.86	37.5	300	UP	UP	.43	.15	.07	.18
Cruise ( $M_{MO}$ )	352	.90	26.5	300	UP	UP	.55	.21	.11	.24
Dive ( $M_D$ )	412	.95	21.5	350	UP	UP	.53	.25	.13	.28
Dive ( $M_D$ )	258	.95	42	300	UP	UP	.41	.17	.11	.18
Cruise (1.4 $V_S$ )	221	.74	38	300	UP	UP	.22	.14	.05	.15
Cruise	216	.435	15	308.5	UP	UP	.33	.13	.11	.15
Descent	246	.45	10	308.5	UP	UP	.49	.13	.12	.17
Holding	256	.4	1.5	308.5	UP	UP	.50	.13	.13	.16
Holding	160	.292	10	308.5	DOWN	UP	.29	.10	.08	.13
Approach (1.3 $V_S$ )	139	.21	0	308.5	DOWN	DOWN	.26	.10	.09	.12
Landing (1.3 $V_S$ )	133	.2	0	308.5	DOWN	DOWN	.24	.09	.09	.12
Landing (1.3 $V_S$ )	141	.213	0	348	DOWN	DOWN	.21	.09	.06	.12
Landing (DLC ON)	133	.2	0	308.5	DOWN	DOWN	.26	.09	.10	.12
Landing (1.4 $V_S$ )	143	.262	10	308.5	DOWN	DOWN	.21	.09	.05	.12

TABLE 3. AUTOMATIC LANDING SYSTEM MAJOR ELEMENTS

<u>Item</u>	<u>No. Req.</u>	<u>Remarks</u>
Pitch Computer	2	Each computer is dual channel.
Roll Computer	2	
Yaw Computer	2	
Roll A/P Servo	2	Each servo is in-line monitored.
Pitch A/P Servo	2	
Yaw A/P Servo	2	
Aileron Position Sensor	2	Each sensor is dual
Rudder Position Sensor	2	Each sensor is dual.
Yaw Rate Gyro	3	Each has limited in-line monitoring.
Mode Annunciator	2	
Warning Indicator	2	
Mode Select Panel	1	5 Modules
Normal Accelerometer	3	
Lateral Accelerometer	3	
Attitude Gyro	3	Each has limited in-line monitoring
Radio Altimeter	2	Each has dual outputs with high integrity monitoring.
ILS Receiver	2	Computer is dual channel
Speed Control Computer	1	Servo is in-line monitored
Autothrottle Servo	1	
Longitudinal Accelerometer	2	
FCES Computer	1	Provide for fail-op/fail-pass DLC
DLC Servo	2	Each is in-line monitored
Trim Augmentation Computer	1	Provides for fail-op/fail-pass auto pitch trim.
Angle of Attack Sensor	2	Each has limited in-line monitoring.
Air Data Computer	2	Each has limited in-line monitoring.
Altimeter	2	
IAS/M Indicator	2	
VSI	2	
ADI	2	
HSI	2	
Radio Altitude Indicator	2	
Compass System	2	
Hydraulic Source	2	
Electric Source	3	

TABLE 4. ESTIMATED MTBF'S VS MTBF LOWER LIMITS

<u>Item</u>	<u>MTBF Lower Limit</u>	<u>Latest MTBF Point Est.</u>	<u>Mature MTBF</u>
Roll Servo	4,000	14,000	20,000
Pitch Servo	10,000	40,000	49,000
Roll Computer	1,500	1,800	3,350
Pitch Computer	1,500	2,900	2,700
Yaw Computer	2,000	3,100	4,600
ILS Receiver	2,000	3,300	3,500
Radio Altimeter	2,000	2,700	3,500
Vertical Gyro	1,800	2,900	3,720
Lateral Accelerometer	32,000	**	63,500
Normal Accelerometer	32,000	**	63,500
Warning Indicator	10,000	14,600	48,000

\*\*No reported failures in 400,000 accelerometer flight hours



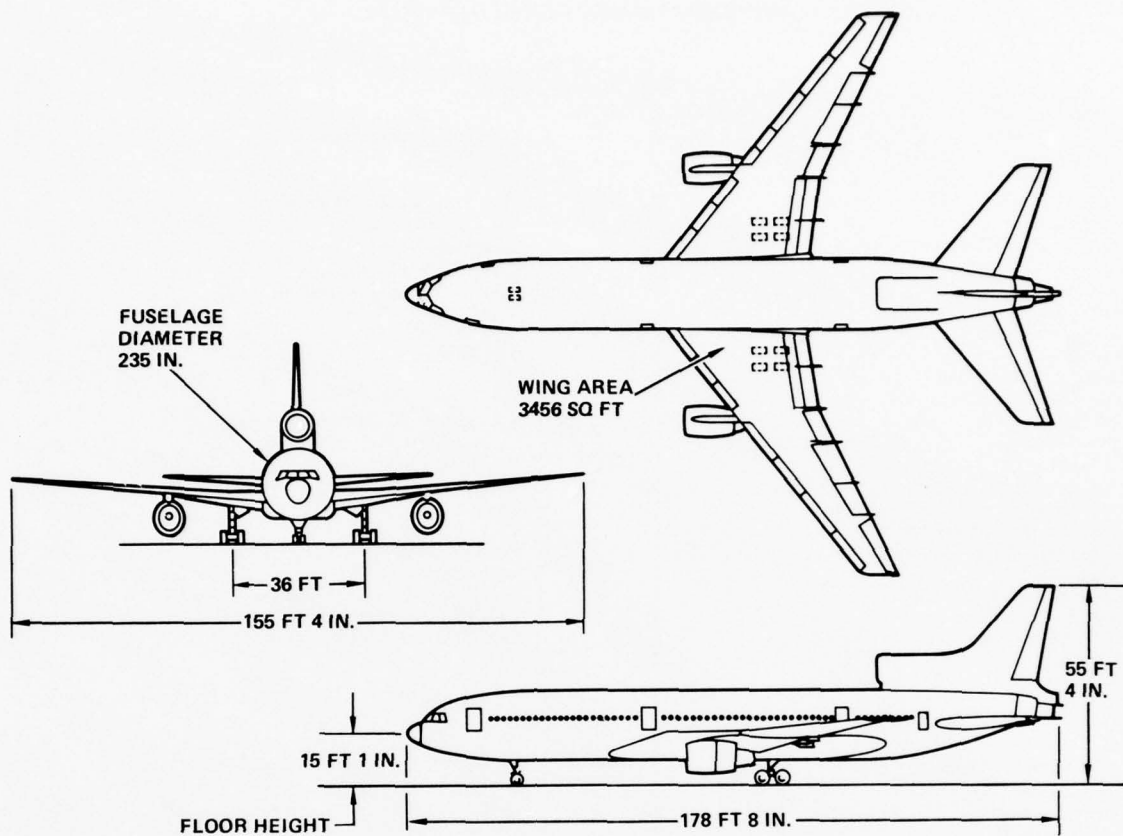


FIGURE 1. L-1011 GENERAL ARRANGEMENT

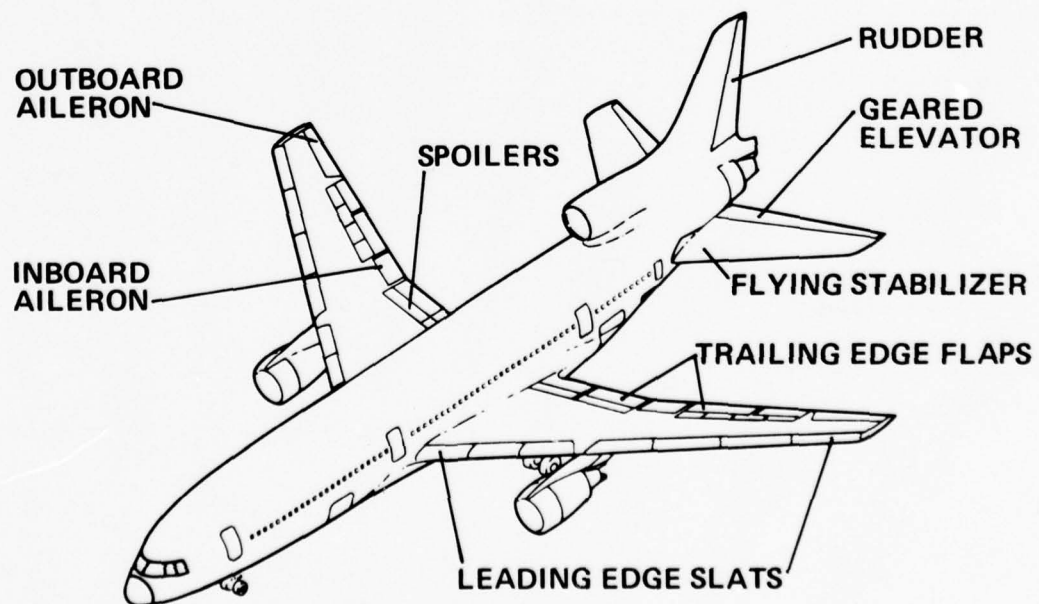


FIGURE 2. CONTROL SURFACE ARRANGEMENT

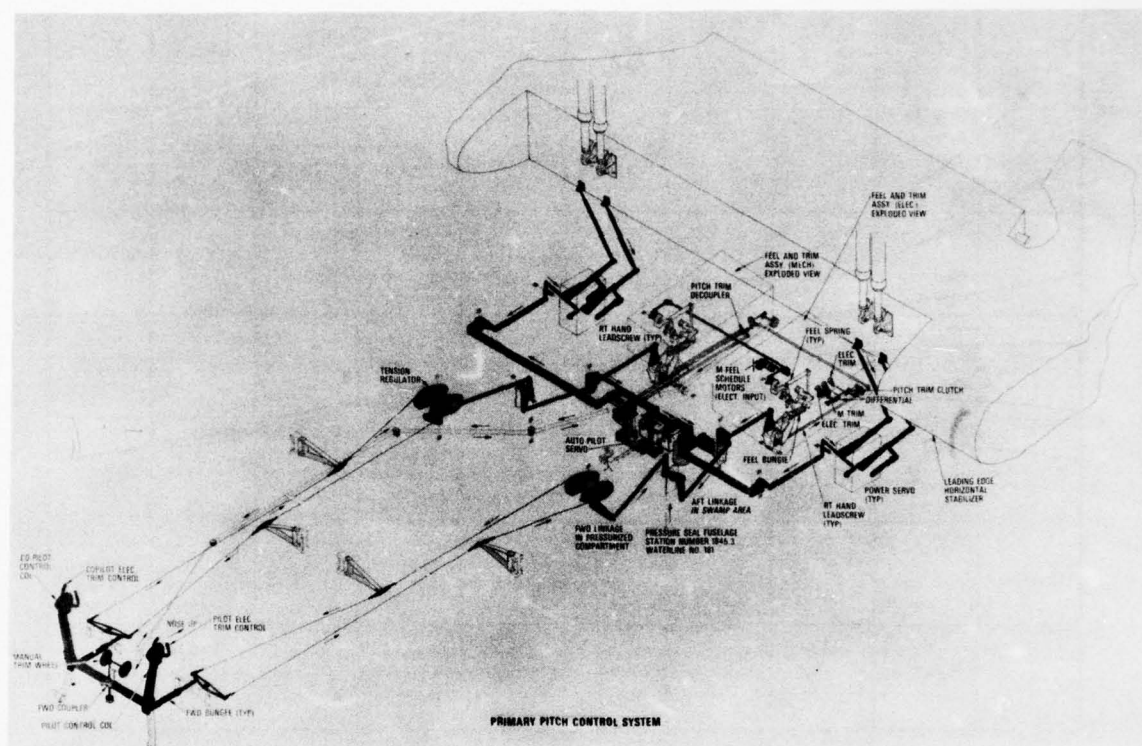


FIGURE 3. PRIMARY PITCH CONTROL SYSTEM

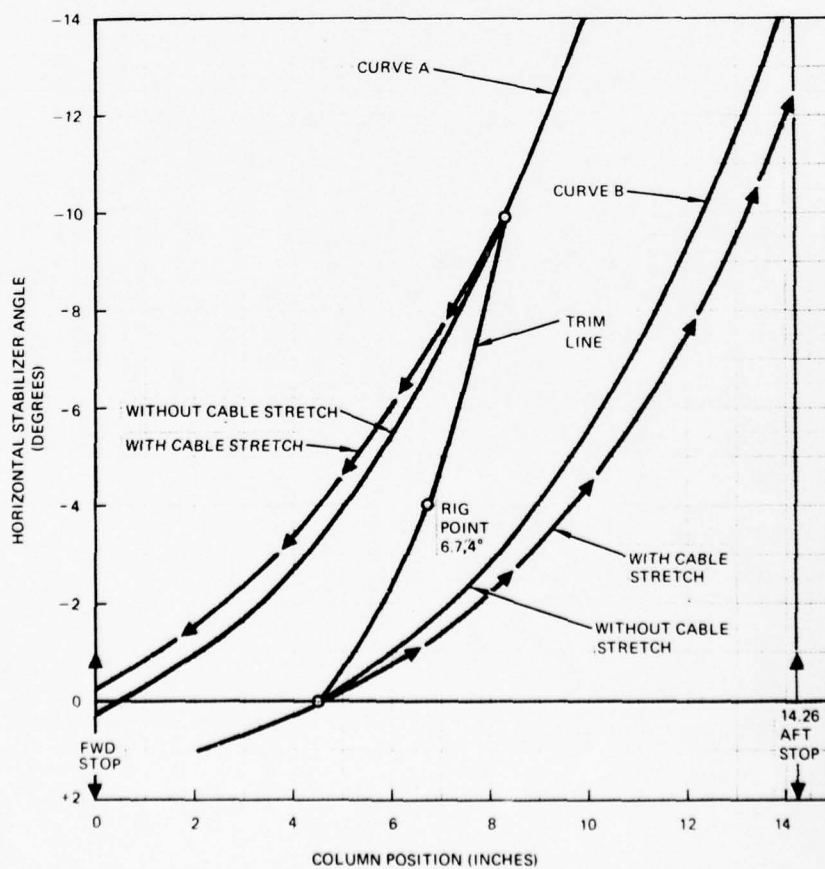


FIGURE 4. COLUMN TO HORIZONTAL STABILIZER DEFLECTION RELATIONSHIP

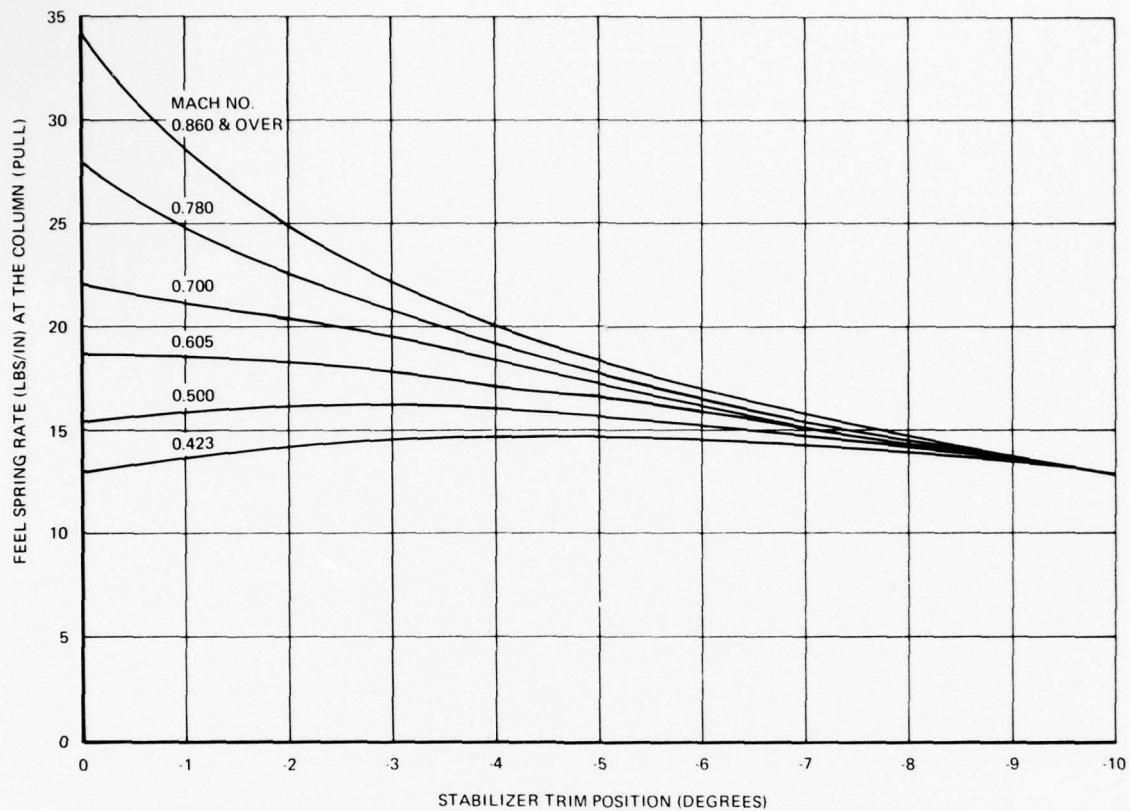


FIGURE 5. PITCH FEEL SPRING RATE AT COLUMN VS. HORIZONTAL STABILIZER TRIM ANGLE (CABLE FLEXIBILITY INCLUDED)

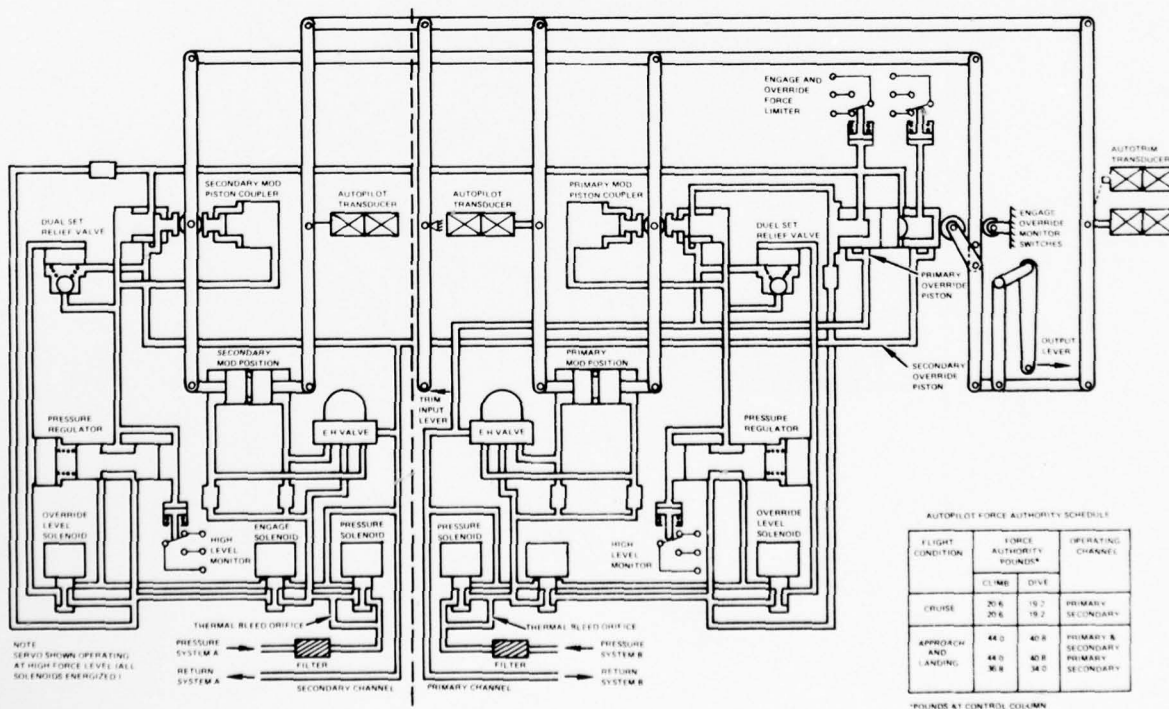


FIGURE 6. STABILIZER AUTOPILOT SERVO SYSTEM SCHEMATIC

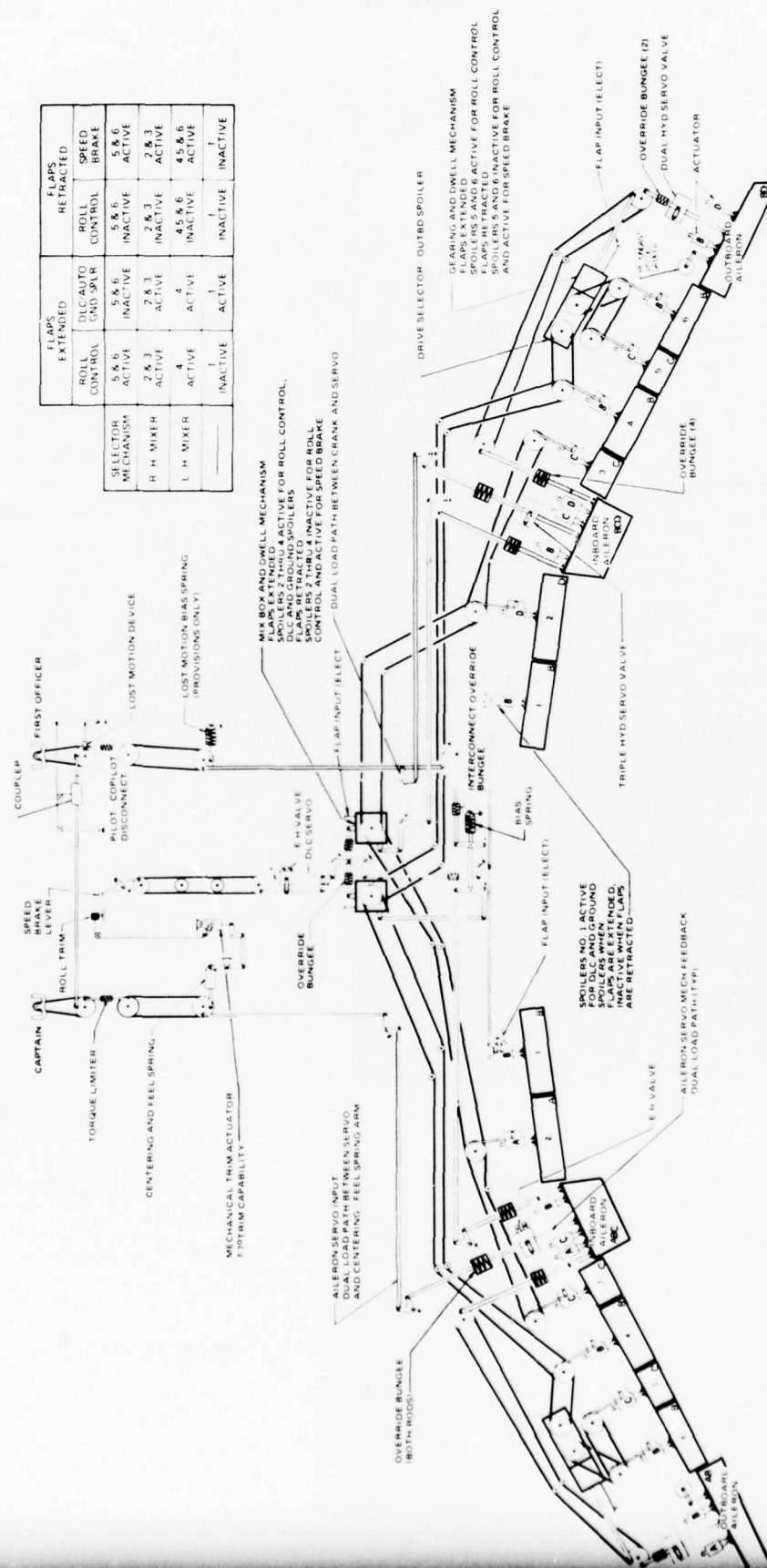
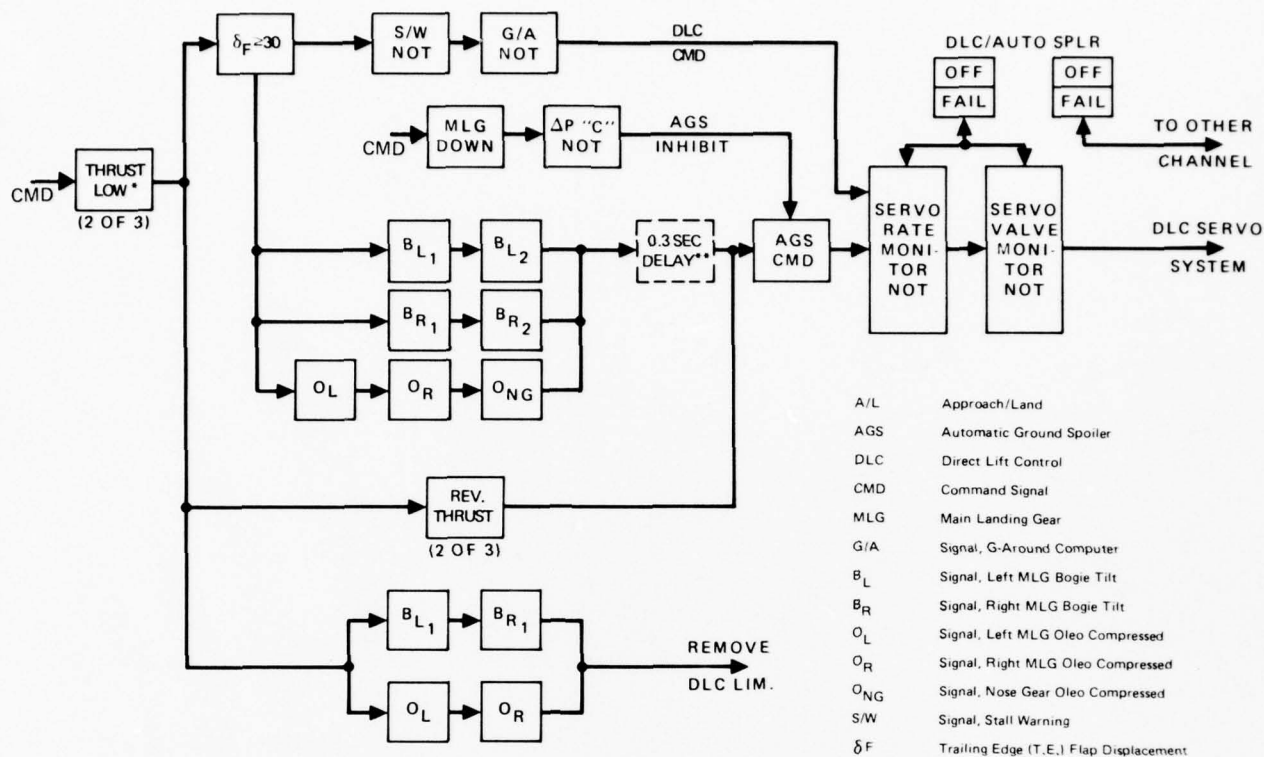


FIGURE 7. ROLL/DLC CONTROL SYSTEM SCHEMATIC





\*OR A/L TRACK

\*\*INHERENT IN SYSTEM

FIGURE 8. DLC AND AUTOMATIC SPOILER DEPLOYMENT SYSTEM INTERLOCK

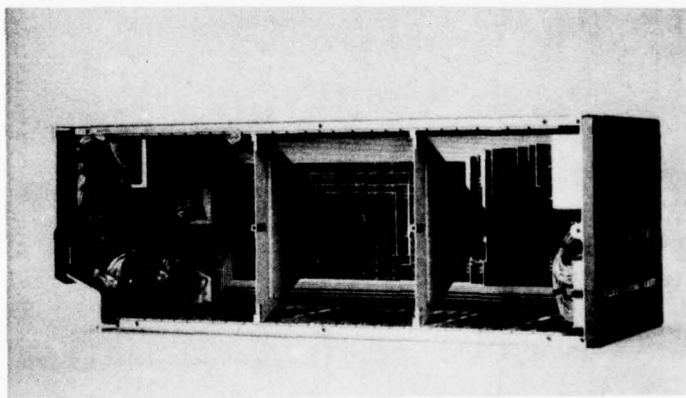
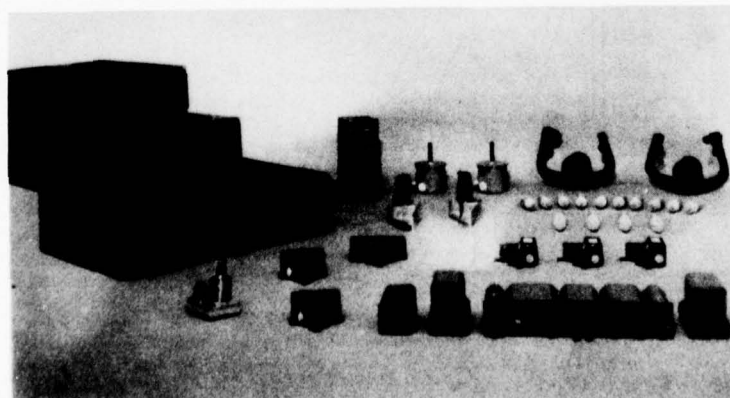
FIGURE 9A. THE COLLINS FCS-110, L-1011  
AFCS HARDWARE

FIGURE 9B. PITCH COMPUTER CHASSIS

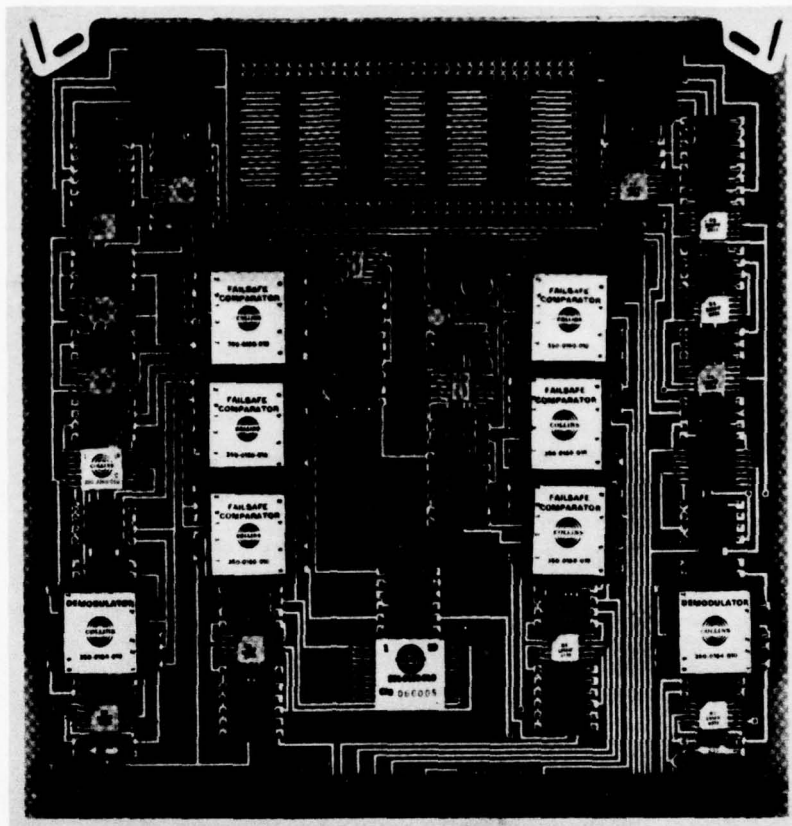


FIGURE 9C. PRINTED CIRCUIT CARD

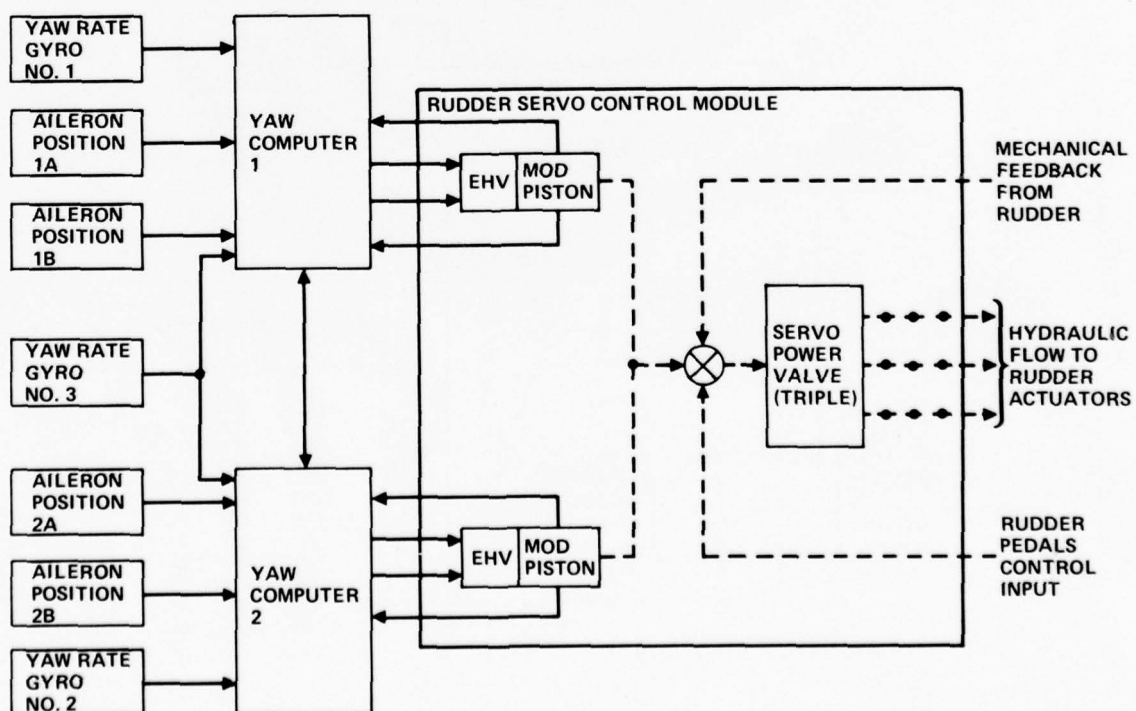


FIGURE 10. STABILITY AUGMENTATION SYSTEM-CRUISE MODE

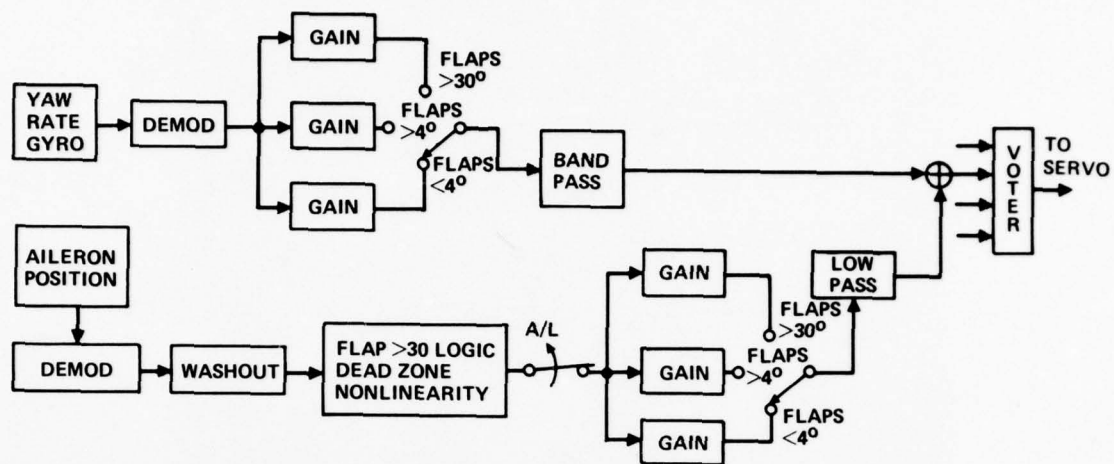


FIGURE 11. YAW SAS CRUISE MODE COMPUTATION

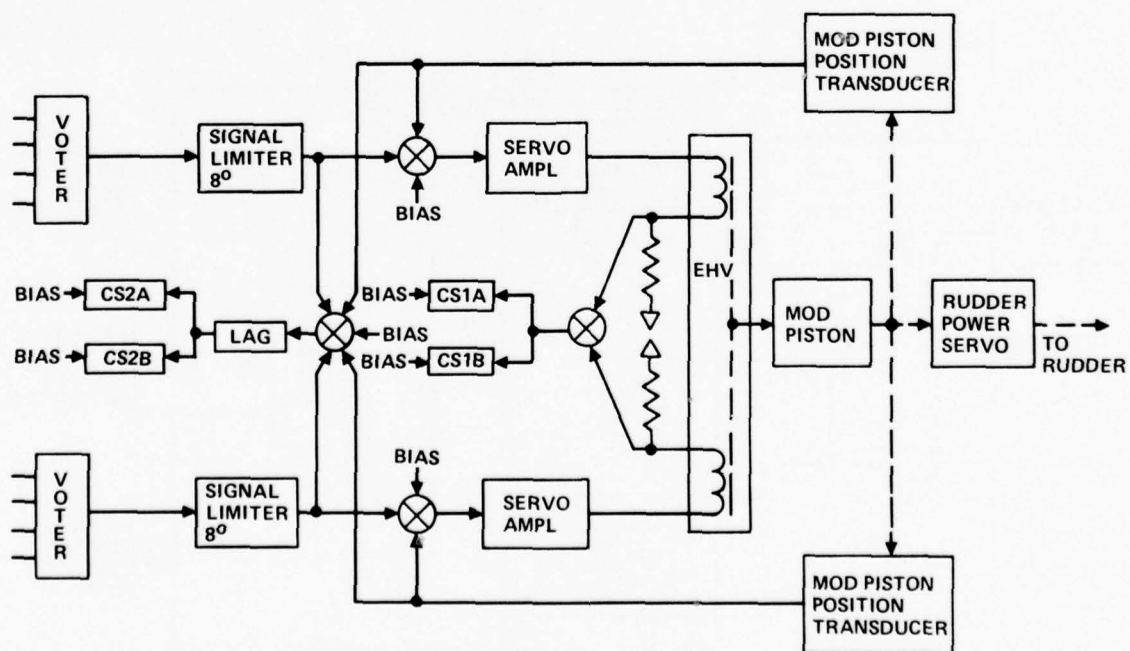


FIGURE 12. YAW SAS SERVO LOOP





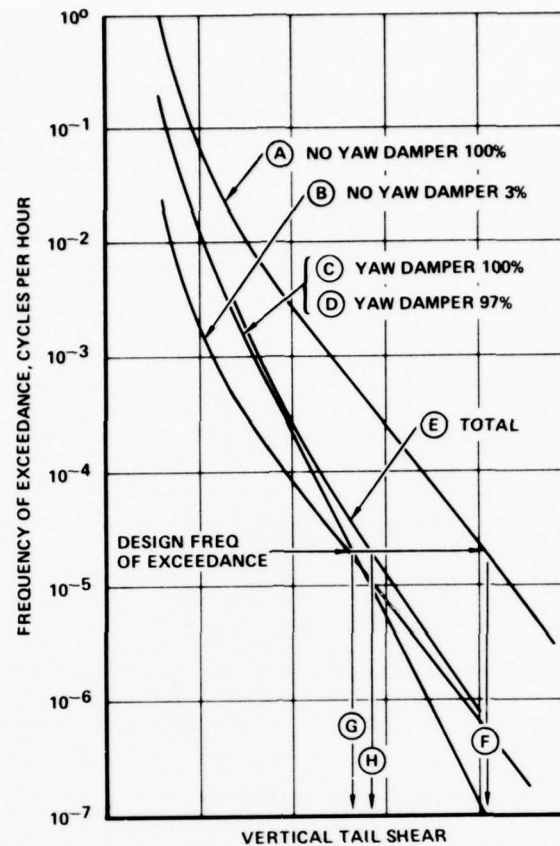


FIGURE 15. FREQUENCY OF EXCEEDANCE OF VERTICAL TAIL SHEAR WITH AND WITHOUT YAW DAMPER

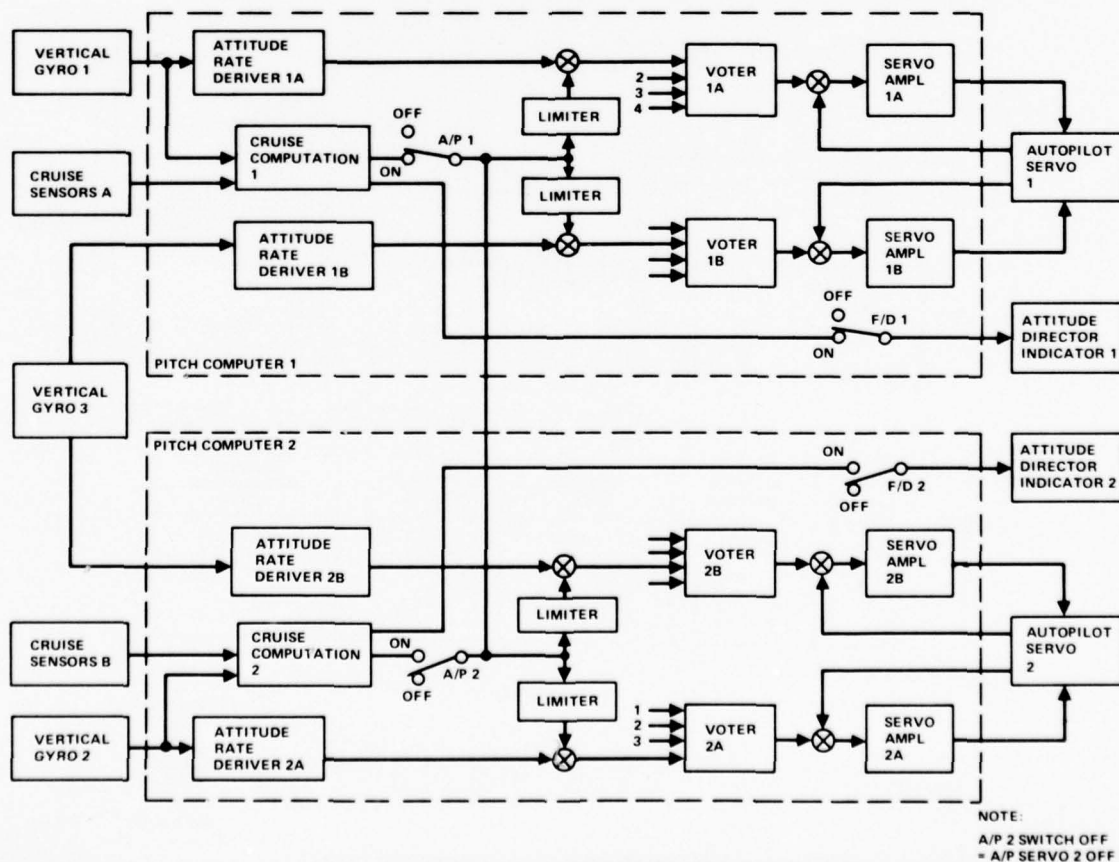


FIGURE 16A. APFDS CRUISE CONFIGURATION (PITCH AXIS)

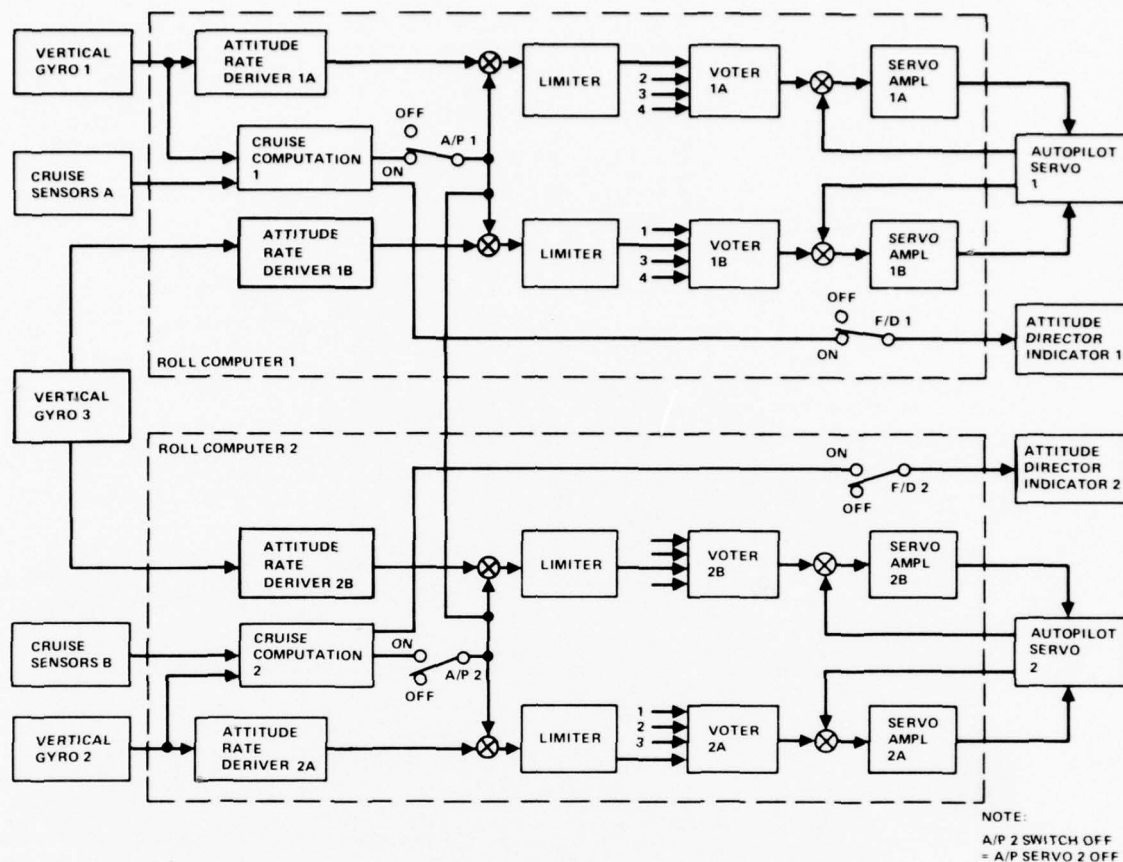


FIGURE 16B. APFDS CRUISE CONFIGURATION (ROLL AXIS)

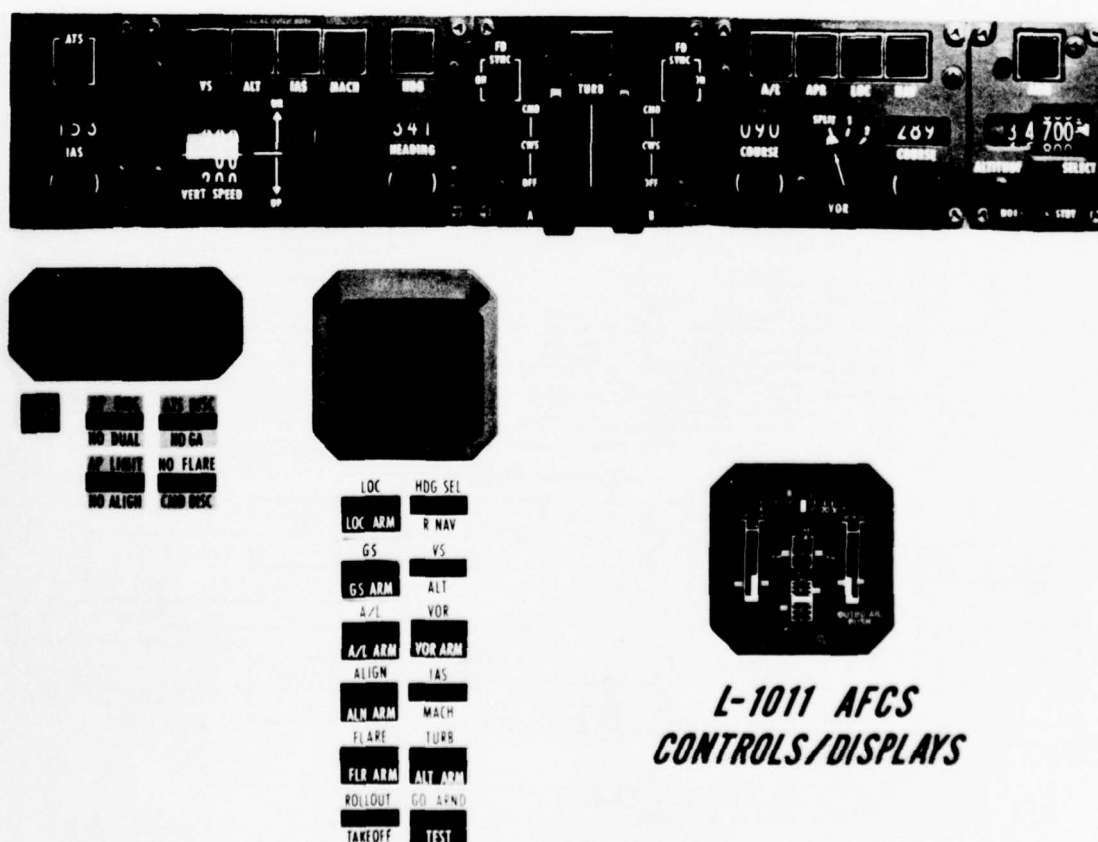


FIGURE 17. AFCS CONTROLS AND DISPLAYS

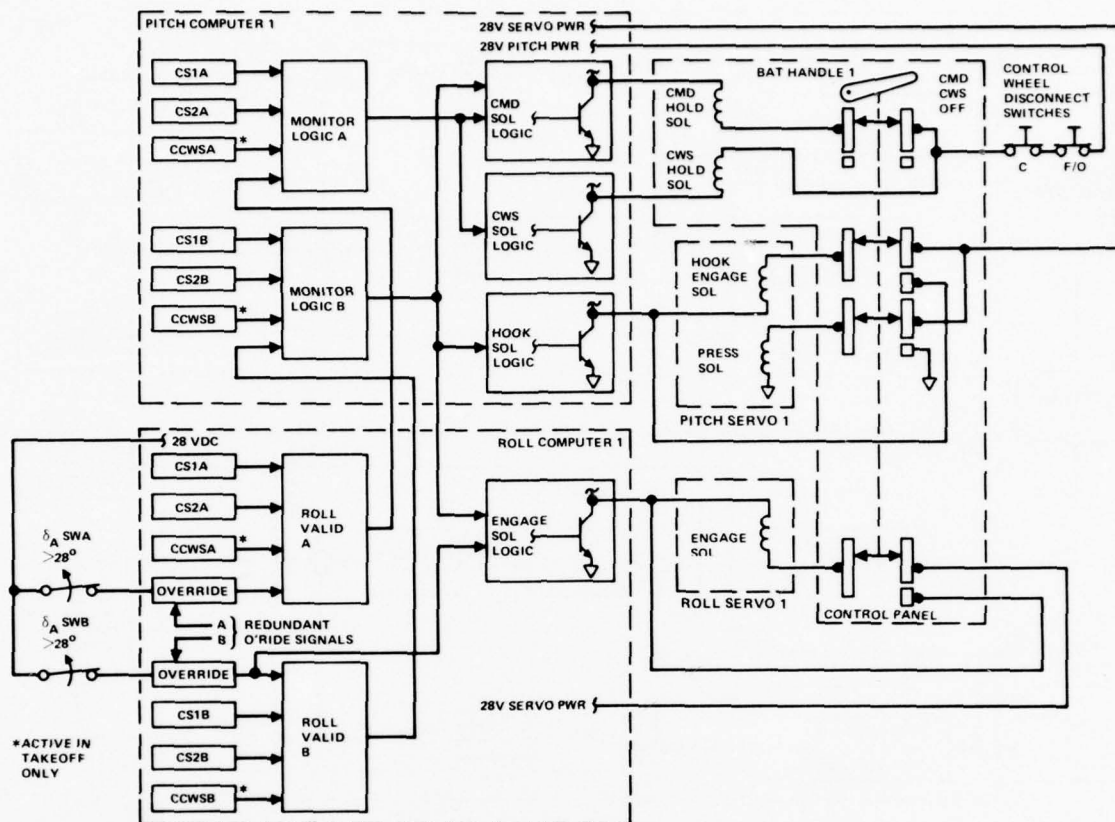


FIGURE 18. PITCH AND ROLL ENGAGE/DISENGAGE LOGIC

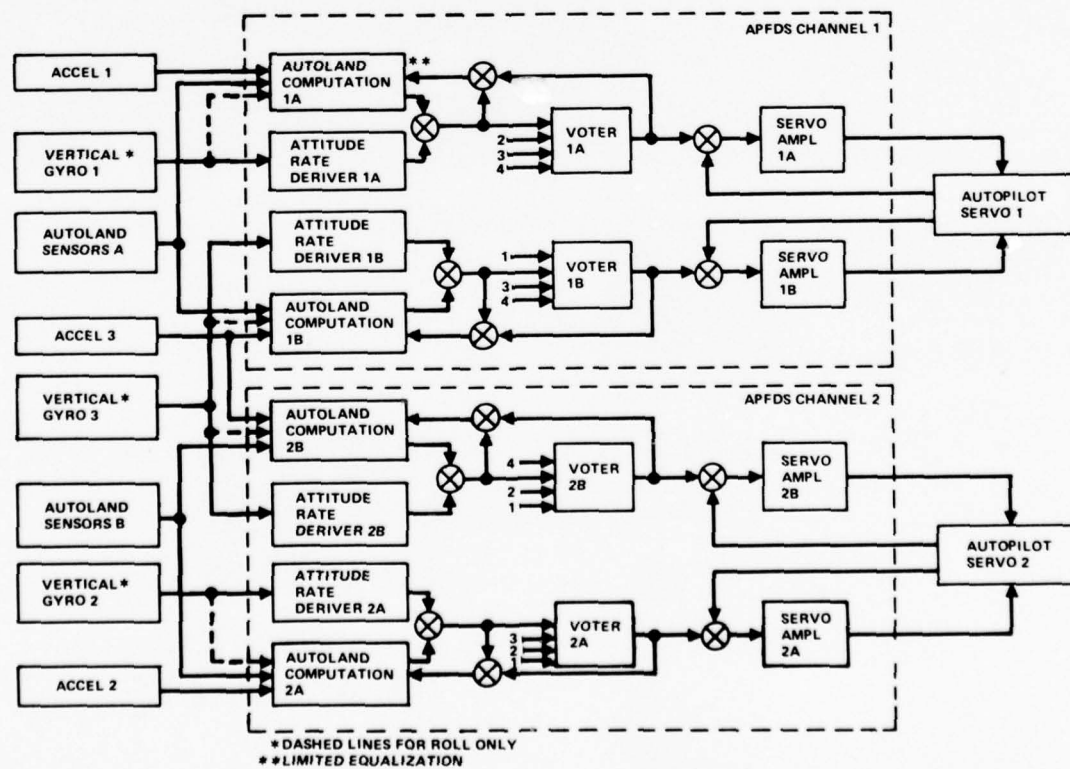


FIGURE 19. APFDS APPROACH/LAND MODE CONFIGURATION

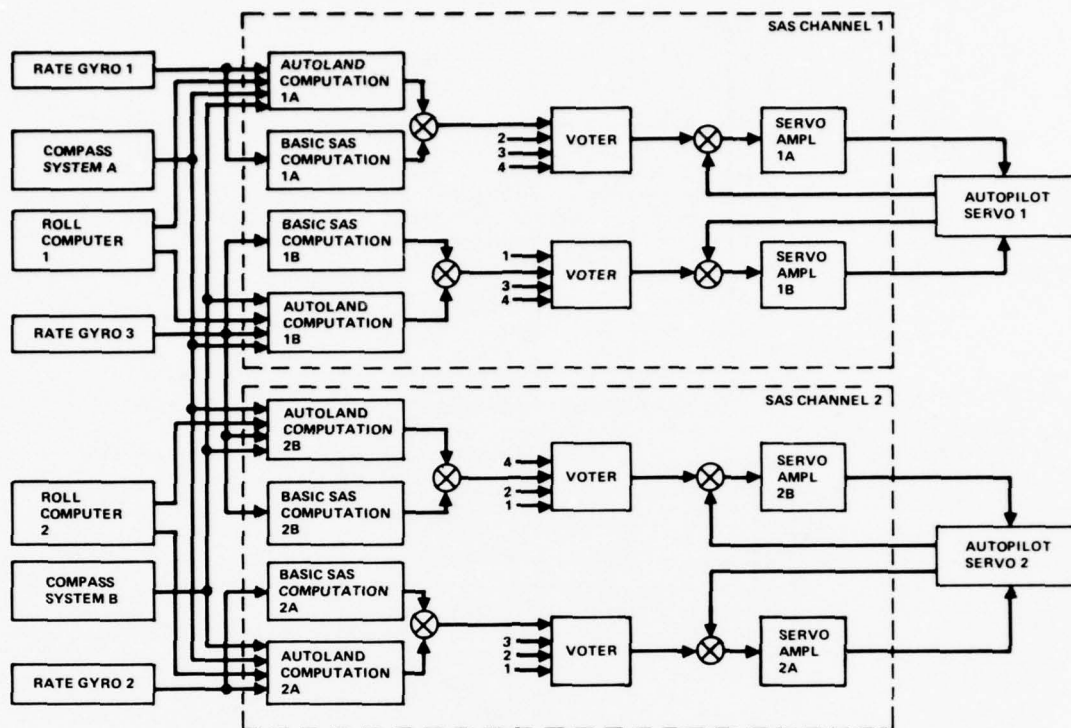


FIGURE 20. SAS APPROACH/LAND MODE CONFIGURATION

- ① THE INDIVIDUAL SPOILER POWER SERVOS AND DRIVE LIKAGES ARE NOT ILLUSTRATED
- ② POSITION IS ACTUALLY MEASURED RELATIVE TO TRIM AS DEFINED BY THE PITCH TRIM ACTUATOR POSITION

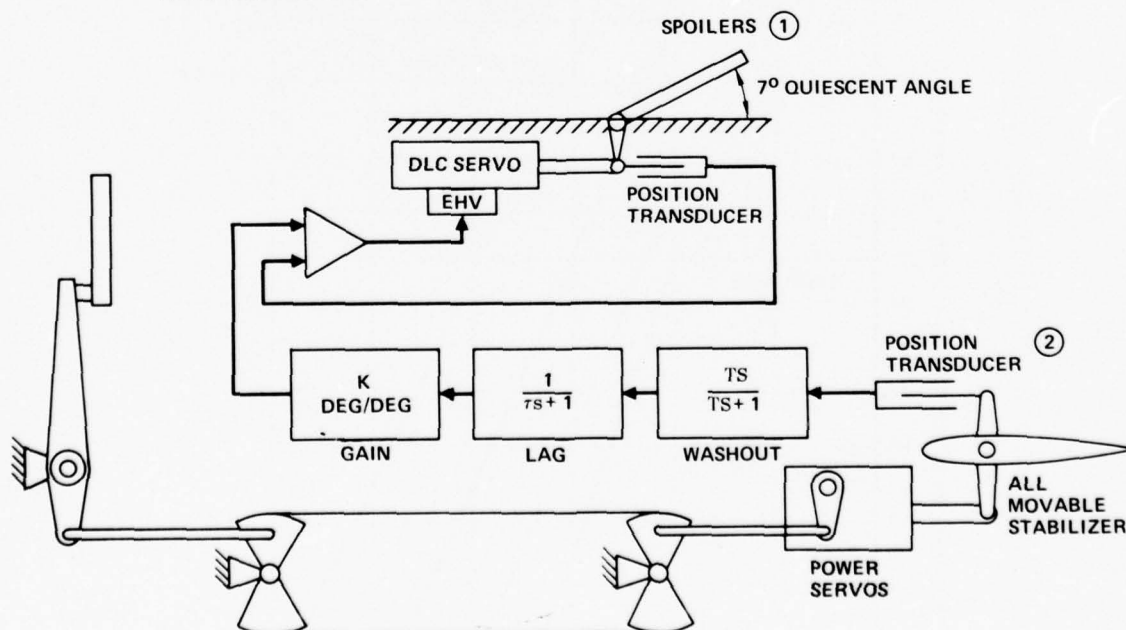


FIGURE 21. BASIC DLC MECHANIZATION SCHEME



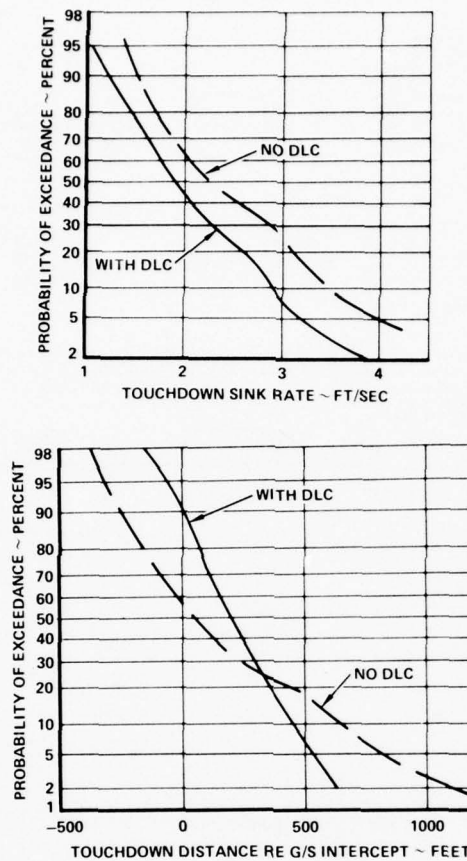
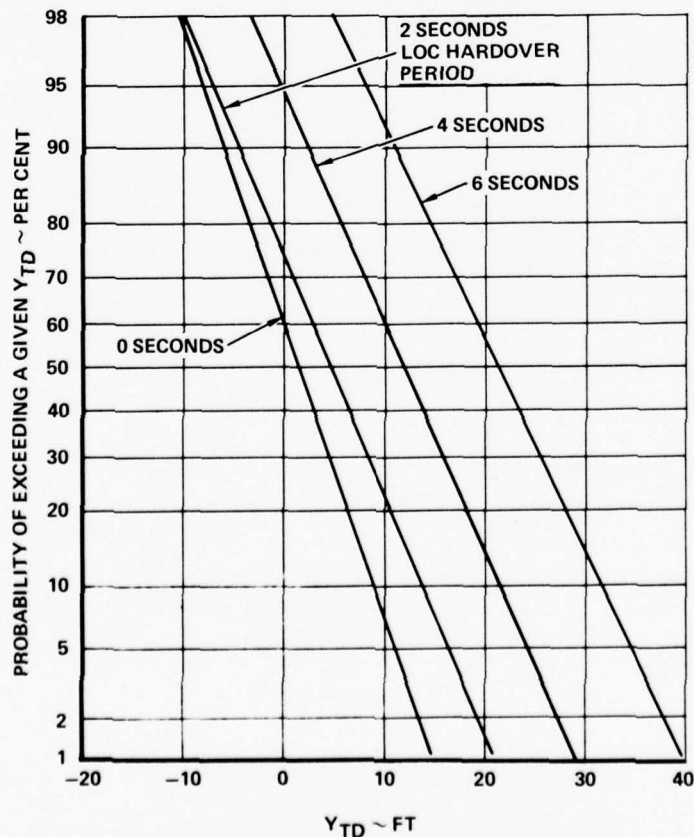


FIGURE 22. LONGITUDINAL TOUCHDOWN PERFORMANCE WITH AND WITHOUT DLC

FIGURE 23. LATERAL DEVIATION AT TOUCHDOWN WITH FAA WIND MODEL AT 15 KNOT CROSSWIND,  $2.5 \mu a(1 \text{ SIGMA})$  RADIO NOISE,  $187 \mu s$  LOC HARDOVER IN DIRECTION OF WIND INITIATED AT 50 FEET BEFORE TOUCHDOWN.

## LES COMMANDES DE VOL DE CONCORDE

par

MM. M. BOSSARD et R. DEQUE  
Ingénieurs au Bureau D'Etudes de la Division Avions  
AEROSPATIALE  
316, route de BAYONNE, B.P. 3153  
TOULOUSE (FRANCE)

### RESUME

L'avion de transport supersonique CONCORDE, actuellement en service, est piloté normalement par l'intermédiaire de commandes de vol électriques redondantes. Une commande mécanique de secours est disponible pour assurer le pilotage de l'avion dans les cas de perte des deux commandes électriques. Après une description du système, ses performances principales sont présentées. Ces commandes de vol présentent un degré de sécurité très élevé ; nous exposons les méthodes mises en oeuvre pour l'obtenir. L'AEROSPATIALE étudie en Laboratoire, et prochainement en vol, des systèmes de commandes de vol entièrement électriques basés sur l'utilisation de techniques digitales qui pourraient trouver leur application sur un avion de transport supersonique de 2e génération. Une description sommaire des solutions étudiées est donnée.

### 1. INTRODUCTION

Une étape importante dans le domaine de la vitesse du transport aérien civil a été franchie par CONCORDE grâce aux progrès importants effectués dans différents domaines techniques (aérodynamique, propulsion, structure, etc...)

Ceci est aussi vrai en ce qui concerne les commandes de vol ; en effet dès le début du projet, il est apparu que, compte tenu des contraintes résultant du domaine de vol, des formes aérodynamiques et des performances de cet avion, des commandes de vol électriques s'imposaient par leurs performances pour obtenir de bonnes qualités de vol. L'expérience en essais en vol et maintenant en exploitation commerciale, a parfaitement justifié les choix effectués. Le manque d'expérience suffisante des commandes de vol électriques même sur avions militaires, a conduit à installer une commande mécanique de secours. L'étape suivante en cours d'étude à l'AEROSPATIALE, consiste à s'affranchir des contraintes liées à cette commande mécanique en masse, complexité et performance par l'utilisation de commandes de vol entièrement électriques, qui permettront en particulier de tirer parti des avantages liés à l'application des concepts CCV.

### 2. DESCRIPTION

#### 2.1. COMPOSITION (voir figures 1 et 2)

Les commandes de vol assurent le contrôle des 6 éleveurs de bord de fuite et des 2 gouvernes de direction, les 6 éleveurs et ces 2 gouvernes constituant la totalité des gouvernes de CONCORDE. Les commandes de vol comprennent :

- des servocommandes de puissance électrohydrauliques
- une commande électrique dite bleue qui est la commande normale
- une commande électrique dite verte, totalement distincte mais identique à la commande bleue
- une commande mécanique de secours
- un ensemble mécanique, hydraulique et électrique, utilisé dans les 3 modes de commande précédents et comprenant les organes de pilotage, leur conjugaison et le système de restitution d'efforts
- une commande électrique "en efforts" utilisable en cas de blocage mécanique des organes de pilotage
- un système de stabilisation artificielle de l'avion
- un système de protection contre les incidences excessives
- un trim électrique servant notamment à la restitution de stabilité statique.

#### 2.2. SERVOCOMMANDES DE PUISSANCE (voir figure 3)

Chaque gouverne est actionnée par une servocommande double corps en tandem, à corps mobile. Par clapet navette, chaque corps peut être alimenté par l'un des 2 circuits hydrauliques.

Chaque corps comprend :

- un distributeur, assurant l'alimentation hydraulique de chaque chambre,
- une servovalve
- une électrovanne assurant l'alimentation hydraulique de la servovalve précédente
- un ensemble hydromécanique de stabilisation de la servocommande.

Les distributeurs des 2 corps sont liés mécaniquement l'un à l'autre.

#### 2.3. COMMANDE ELECTRIQUE BLEUE (voir figure 4)

Elle est composée de plusieurs chaînes distinctes assurant respectivement le contrôle :

- des éleveurs internes
- des éleveurs médians et externes
- des gouvernes de direction.

Chaque chaîne est formée de détecteurs inductifs de position d'organe de pilotage et de gouverne, d'ampli d'asservissement et d'un ensemble électrique de surveillance.

Les servocommandes sont contrôlées à partir d'une des 2 servovalves mentionnées précédemment. En cas de panne d'une chaîne, l'alimentation hydraulique des servovalves correspondantes est coupée et la chaîne bleue est remplacée par son homologue verte.

Notons encore que l'ensemble des chaînes bleues est alimenté en courant 1 800 Hz par un convertisseur statique qui lui est propre. Un convertisseur analogue alimente l'ensemble des chaînes électriques vertes.

#### 2.4. COMMANDE MECANIQUE (voir figures 5 et 6)

Par câbles dans le fuselage et bielles dans la voilure et la dérive, elle comporte comme particularités :

- des vérins-relais de profondeur, gauchissement et direction, situés dans la pointe avant et destinés d'une part à éviter que les frottements de la commande mécanique soient ressentis par les pilotes, d'autre part à entraîner les organes de pilotage automatique
- des mélangeurs mécaniques assurent l'addition des ordres de profondeur et de gauchissement des élevons.

#### 2.5. ORGANES DE PILOTAGE ET SYSTEME DE RESTITUTION D'EFFORTS

Les organes de pilotage (manche, volant, pédales) sont de forme classique. Côtés pilote et copilote sont liés mécaniquement et entraînent par une liaison doublée les détecteurs inductifs des commandes électriques bleue et verte et la commande mécanique de secours.

Le système de restitution d'efforts (voir figure 7) introduit au niveau de cette liaison des efforts dépendant du braquage, du mach, de la vitesse corrigée et du centrage. Les efforts sont obtenus par une bielle à ressort et l'asservissement en pression de vérins électrohydrauliques. Les informations nécessaires au calcul de la pression sont fournies par les centrales anémométriques aux 2 calculateurs du système qui assurent le calcul de la pression, l'asservissement et la surveillance des vérins. Aucune panne simple électrique ou hydraulique ne modifie les efforts ressentis aux organes de pilotage.

L'annulation des efforts s'effectue à position constante des organes de pilotage. En profondeur cette annulation peut être obtenue par l'une des deux commandes électriques ou en secours par une commande mécanique. En gauchissement et direction, la commande d'annulation d'efforts est purement mécanique.

#### 2.6. COMMANDE EN EFFORTS

Les manches pilote et copilote sont équipées d'un moyeu dynamométrique. En cas de blocage des manches ou des volants, après engagement du système par le pilote, les efforts exercés au niveau d'un volant sont transformés en ordres électriques par le moyeu. Ces ordres, reçus par les calculateurs de protection hautes incidences (Cf. ci-après), commandent le braquage des élevons.

#### 2.7. STABILISATION ARTIFICIELLE (voir figure 8)

Ce système est entièrement doublé. Chaque demi-système comprend :

- des gyromètres de tangage, roulis et lacet
- 1 accéléromètre d'accélération transversale
- 1 calculateur

Outre les informations des gyromètres et accéléromètres, les calculateurs reçoivent des informations de Mach et d'incidence.

Chaque demi-système fournit aux commandes de vol électriques, des ordres de stabilisation de l'avion, d'amélioration du temps de réponse et de coordination de virage. De plus, par détection accélérométrique, il atténue le dérapage transitoire consécutif à une panne de réacteur en provoquant un braquage automatique de la gouverne de direction.

Les calculateurs sont du type doublé auto-surveillé ; chacun peut alimenter indifféremment les chaînes électriques bleues et vertes. Les pannes des gyromètres et des accéléromètres, ainsi que la majeure partie des pannes des calculateurs, ne provoquent pas de commutation des chaînes électriques bleue ou verte.

#### 2.8. PROTECTION CONTRE LES INCIDENCES EXCESSIVES

Le système génère :

- à incidence relativement faible, des vibrations au niveau du manche (dispositif classique)
- à incidence plus élevée, une alarme inmanquable constituée d'efforts alternés violents et rapides au niveau du manche. Ces efforts n'existent que lorsque le manche est en position cabré.

- un déroulement du trim fonction de l'incidence

Il commande aussi lorsque nécessaire :

- une augmentation des ordres des stabilisateurs de profondeur et de lacet
- une déconnexion du pilote automatique et une inhibition du trim automatique.

Le système comporte essentiellement deux calculateurs analogiques autosurveillés. Outre les fonctions de protection contre les incidences excessives, ces calculateurs assurent le traitement des ordres de pilotage en effort (Cf. § 6. ci-avant).

#### 2.9. TRIM ELECTRIQUE

De manière classique, les calculateurs de trim électrique (au nombre de deux, auto-surveillés) commandent :

- l'annulation des efforts en vol stabilisé en pilotage manuel selon les ordres pilote, en pilotage automatique automatiquement

- la restitution de la stabilité statique réglementaire en efforts.

La réalisation des ordres est assurée par 2 moteurs électriques (un en fonctionnement, l'autre en attente),

surveillés, qui modifient la liaison cinématique entre manches d'une part, bielle à ressort et vérins de restitution d'efforts, d'autre part (Cf. § 5 ci-avant).

### 3. PERFORMANCES

La définition des commandes de vol du CONCORDE résulte d'exigences de performances particulièrement sévères liées aux caractéristiques de l'avion et à son domaine d'utilisation. On peut citer en particulier

- un domaine de vol particulièrement large : voir planche 9 les plages de variation d'altitude et de mach entraînant des variations importantes de caractéristiques de l'avion (voir fig. 10 l'évolution des braquages par g)
- une large variation des marges statiques suivant les centrages et les conditions de vol (Voir fig 19).

#### 3.1. PERFORMANCE DES CHAINES DE COMMANDE DE GOUVERNES

Ceci nous a conduit à imposer des conditions de précision de commande d'élévons (hystérésis =  $\pm 5'$ ) qui ont pu être atteintes grâce à la commande électrique. En commande mécanique, on a dû accepter des précisions dégradées (hystérésis =  $\pm 24'$ ) qui permettent d'assurer le pilotage de l'avion dans le domaine de vol normal dans des conditions de confort dégradées et avec des consignes particulières notamment sur les centrages à respecter. Les dimensions de l'avion et son inertie sont telles que les exigences sur les performances dynamiques sont en apparence relativement modestes, mais n'ont pu être satisfaites que grâce à des dispositifs spéciaux de stabilisation de servo-commande :

- bande passante des servo-commandes : déphasage à 2 Hz en électrique  $< 45^\circ$
- vitesse Max sans charge aérodynamique : 28 à 40 °/s sur servo-commande d'élévons, 46 à 51 °/s sur servo-commande de direction.

#### 3.2. PERFORMANCE DES SYSTEMES AUTOMATIQUES D'AIDE AU PILOTAGE

Les systèmes d'aide automatique au pilotage ont été initialement définis en fonction des caractéristiques de l'avion et des divers critères de qualités de vol trouvés dans la littérature. Très tôt dans l'étude de l'avion on a mis en place des moyens de simulation sophistiqués. C'est grâce à ces moyens qui ont permis de faire intervenir les jugements des pilotes que l'on a pu définir de façon satisfaisante ces divers systèmes. Cette mise au point des systèmes était telle au 1er vol, qu'ils ont pu être engagés dès le premier décollage, et leur définition a assez peu évolué par la suite, sauf dans certains cas que nous mentionnerons ci-après.

##### 3.2.1. SYSTEME DE RESTITUTION ARTIFICIELLE D'EFFORTS

Leur définition résulte d'un compromis entre le souhait des pilotes d'avoir à développer des efforts faibles dans les manoeuvres normales de pilotage peu variables avec les conditions de vol et de chargement, et d'autre part de satisfaire pour la profondeur et la direction aux exigences suivantes :

PROFONDEUR : Effort à 1,6 g  $\leq 230$  N  
2,5 g  $\geq 230$  N

DIRECTION : Effort  $> 900$  N pour les braquages limites tolérés par la résistance structurale.

Ces compromis n'ont pu être atteints pour la direction que par l'introduction d'un 2e seuil d'effort fonction des conditions de vol un peu avant le braquage limite toléré par la structure.

En gauchissement la structure permet les débattements maxi de gouverne.

Pour la profondeur, ces exigences ont conduit à des efforts par g compris entre 200 et 300 N/g (voir fig 12)

##### 3.2.2. SYSTEME DE TRIM

En plus des fonctions usuelles de trim d'annulation d'effort, le système de trim a pour fonction de réaliser une stabilité statique manche libre artificielle. En effet le CONCORDE, en dehors de l'instabilité statique en transsonique, présente une stabilité statique voisine de l'indifférence (légèrement positive ou légèrement négative) dans la majeure partie des phases de vol. Ceci, bien que ne déplaçant pas en général aux pilotes, n'a pas été accepté par les autorités de certification, et il a fallu rétablir une stabilité statique manche libre positive.

D'autre part, pour ne pas dégrader les caractéristiques de pilotage, cette stabilité en effort doit rester toujours modérée. La recherche du compromis a été particulièrement difficile ; en effet cette stabilité dépend de nombreux paramètres (Mach,  $V_0$ , Masse, Centrage) et de leur combinaison, et les prendre tous en compte aurait conduit à une complexité excessive. Le compromis retenu a donné les résultats présentés fig. 13.

##### 3.2.3. SYSTEME D'AUTOSTABILISATION

La principale fonction du système est l'augmentation de l'amortissement des oscillations de courte période en roulis, lacet et tangage. Ce système est à autorité limite pour des raisons de sécurité d'ailleurs plus psychologiques que réelles, étant donné le niveau d'autosurveillance très élevé du système. L'optimisation des gains et constante de temps des autostabilisateurs s'est faite progressivement en fonction essentiellement des jugements des pilotes, d'abord sur simulateur, et ensuite en vol.

On a noté quelques différences entre les jugements portés sur simulateur et en vol :

- Certains pilotes ont observé en vol en approche, une tendance au pompage pilote (période 4 à 5 s) qui, dans les mêmes conditions, n'a jamais pu être observée sur simulateur ; l'optimisation des lois de stabilisateur de tangage pour faire disparaître cette tendance, a donc dû être entièrement effectuée en vol.

- Dès les premiers essais sur simulateur, une possibilité de pompage pilote en latéral a été observée en transsonique, due principalement au fort lacet direct induit par les élévons internes ( $\omega_y / \omega_\phi$  élevé). On a donc joué sur les répartitions de braquages gouverne en gauchissement entre les différents élévons et sur les lois d'autostabilisation pour éliminer le problème. Ce phénomène a été aussi observé en vol, mais avec un degré de sévérité nettement inférieur, ce qui a conduit à des réglages différents de ceux initialement proposés sur simulateur.



L'autostabilisateur de tangage a aussi été utilisé pour amortir le 1er mode symétrique de structure fuselage (1,5 à 2 Hz), ce qui n'était pas prévu dans la définition initiale. En effet, les premiers vols effectués en forte turbulence avaient fait apparaître un niveau d'inconfort gênant au poste de pilotage dû à ce mode. L'amortissement du mode a été augmenté artificiellement (au moins double) par un positionnement judicieux du gyromètre et une adaptation du filtrage pour éviter le couplage avec d'autres modes de fréquence plus élevée. En fait c'est la stabilité d'un mode créé par le système (0,8 à 1 Hz) qui limite les gains possibles.

L'influence sur les caractéristiques dynamiques de l'avion du système de stabilisation avec ses réglages de série apparaît sur les figures n° 14 à 17.

### 3.2.4. SYSTEME DE PROTECTION ANTI HAUTES INCIDENCES

Le système a été réglé d'une part pour permettre les évolutions réglementaires aux vitesses d'approche et de décollage (1,5 g à  $V_{REF}$ , 1,4 g à  $V_2$  et 1,6 g à  $V_3$ ) et d'autre part pour éviter que des manœuvres brutales effectuées à ces vitesses, pratiquement application très rapide des braquages maximaux, ne conduisent à excéder l'incidence maximale autorisée de 19,5°.

## 4. SECURITE

### 4.1. OBJECTIFS

- a) aucune panne de probabilité  $> 10^{-5}/h$  de vol ne doit amener une dégradation notable des qualités de vol ou augmentation des charges de travail de l'équipage
- b) les pannes mettant l'avion dans une configuration critique doivent être Extrêmement Rares (probabilité  $< 10^{-7}/h$  de vol)
- c) les pannes mettant l'avion dans une configuration catastrophique doivent être Extrêmement Improbables (probabilité  $< 10^{-9}/h$  de vol).

### 4.2. MISE EN OEUVRE

Le système de commande de vol décrit au § 2 a été dès sa conception étudié pour satisfaire aux objectifs ci-dessus. Pour cela les principes suivants ont été mis en oeuvre :

#### - REDONDANCE

Par exemple chaque gouverne peut être commandée par 3 voies indépendantes. De plus, la plupart des pannes simples n'affectent que les élevons internes ou les élevons externes et médians. Les systèmes automatiques d'aide au pilotage sont doubles.

#### - AUTO-SURVEILLANCE

Chaque chaîne de pilotage ou d'aide automatique de pilotage a la capacité de détecter ses propres pannes avec une probabilité très élevée, de façon à pouvoir en cas de panne passer automatiquement le relais à une chaîne saine. Il faut noter que la commande mécanique de secours ne joue aucun rôle dans la surveillance des chaînes électriques.

#### - PARTITION

Pour pouvoir bénéficier des avantages de la redondance, une ségrégation très poussée est nécessaire aussi bien au niveau des alimentations que des équipements et des cablages. De plus, pour minimiser l'influence des pannes, les chaînes de commandes de vol avec leur surveillance sont séparées par groupe de gouvernes (élevons internes, élevons externes et médians, direction).

#### - LIMITATION D'AUTORITE

Les systèmes automatiques d'aide au pilotage ont leur autorité physiquement limitée au strict nécessaire pour assurer leur fonction.

En plus de ces principes, toutes les règles de l'art ont été appliquées avec beaucoup de soin aussi bien pour les réalisations et installations des éléments mécaniques que pour les éléments électriques ou électroniques.

### 4.3. VERIFICATION

#### 4.3.1. ANALYSE DE PANNES

Les commandes de vol, comme d'ailleurs l'ensemble des systèmes de l'avion, ont fait l'objet d'analyses de pannes très détaillées suivant une méthode développée à l'occasion du programme CONCORDE. Cette méthode consiste à considérer et combiner les pannes des composants ou constituants élémentaires prises chacune avec leur probabilité d'occurrence, pour déterminer la nature et la probabilité des divers types de pannes du système (pannes globales). L'influence des éléments extérieurs au système est aussi prise en compte. Un extrait de la liste des pannes globales est donné figure 18.

#### 4.3.2. ETUDE DES CONSEQUENCES DE PANNES

Les configurations de pannes résultant de l'analyse de sécurité ont été reproduites sur un simulateur de vol comportant la quasi totalité des équipements réels de commandes de vol. Ces pannes ont été introduites dans les différentes conditions de vol de façon à pouvoir faire porter par les pilotes, constructeurs et officiels, un jugement sur la nature des conséquences et vérifier en particulier qu'elles étaient en accord avec les objectifs de sécurité. Bien entendu la validité de la simulation avait auparavant été démontrée par comparaison avec le vol.

A l'issue de ces essais, un certain nombre de ces pannes a été retenu pour essais en vol, soit que le jugement sur simulateur soit difficile (problèmes de pompage pilote par exemple), soit que les conséquences, compte tenu de la probabilité, soient suffisamment importantes pour mériter un jugement en vol. Certaines configurations de pannes, en particulier le pilotage de l'avion en commande mécanique, ont fait l'objet de très nombreux essais en vol.

#### 4.3.3. ESSAIS

Tous les équipements et les systèmes ont été soumis à des essais de qualification sévères : environnement, endurance, fiabilité.

L'ensemble des éléments mécaniques de commandes de vol assemblés sur le banc de commande de vol a déjà subi des essais d'endurance correspondant à 16 000 heures de vol.

Le nombre total d'heures de vol accumulé sur les avions prototypes, présérie et série, sans incident majeur de

commandes de vol, dépasse actuellement 7 000 h. Aucune panne n'a à ce jour nécessité le pilotage de l'avion en commandes mécaniques.

## 5. EVOLUTION

Comme voulu à l'origine, les commandes de vol de CONCORDE permettent le contrôle de l'avion dans tout le domaine de vol en mode mécanique, donc notamment sans stabilisation artificielle. La perte de la commande électrique n'étant pas dangereuse, sa définition est assez simple (très simple même en ce qui concerne la partie purement électrique). Sa certification n'a pas posé de problèmes particuliers, ses composants ont pu être choisis parmi ceux dont les compagnies aériennes ont une bonne connaissance. La contre-partie est assez évidente. D'une part, la cohabitation d'un système normal électrique avec un système de secours mécanique, crée quelques complications. D'autre part, et surtout la volonté de pouvoir piloter l'avion en secours sans l'aide de l'électronique, impose des contraintes dans la définition de l'avion qui doit présenter des qualités de vol "naturelles" satisfaisantes.

Aussi l'AEROSPATIALE, avec l'aide du Service Technique de l'Aéronautique, a-t-elle entrepris l'étude de commandes de vol entièrement électriques.

Nous ne nous étendons pas sur les motivations de telles commandes, motivations souvent exposées.

Rappelons seulement quelques titres :

- possibilité de vol à centrage arrière et donc possibilité d'augmenter la finesse au décollage, de diminuer la traînée d'équilibrage en croisière, de diminuer le dimensionnement de l'empennage horizontal
- possibilité de répartition optimale des charges aérodynamiques de manoeuvre
- possibilité d'amortissement de flutter de voilure ou de modes structuraux de fuselage
- possibilité de réduction de la stabilité de route naturelle de l'avion et donc de la dérive
- facilité de réalisation d'amortisseur de rafales et de modulation de portance.

Ces possibilités sont suffisamment importantes pour que, au moins dans certains programmes, on en tienne compte sur la définition générale de l'avion (on pense que ce sera notamment le cas pour la prochaine génération de transport supersonique). Dès lors, ces possibilités doivent être évaluées aussi précisément que possible avant même le choix d'un programme.

Evaluer signifiera déterminer les possibilités de la technologie, définir une architecture des commandes de vol, apprécier la facilité de pilotage. Dans le domaine de l'aviation civile, la conclusion tangible de cette évaluation sera économique : pour chaque avion défini par sa mission, on devra juger de l'intérêt économique des commandes de vol purement électriques.

Bien sûr, dès lors que ces commandes ont une répercussion importante sur la définition de l'avion, leur prix peut être un peu secondaire. Ce n'est pas une raison néanmoins pour le négliger. On est donc amené à définir les moyens minimaux permettant d'obtenir sécurité et performance.

Dans cette optique, l'AEROSPATIALE, après une première évaluation comparative numérique / analogique, a essayé en laboratoire, plusieurs calculateurs numériques. Une conclusion nette s'est dégagée de ces essais : il existe au moins un calculateur numérique satisfaisant pour réaliser des commandes de vol purement électriques. Satisfaisant signifie ici :

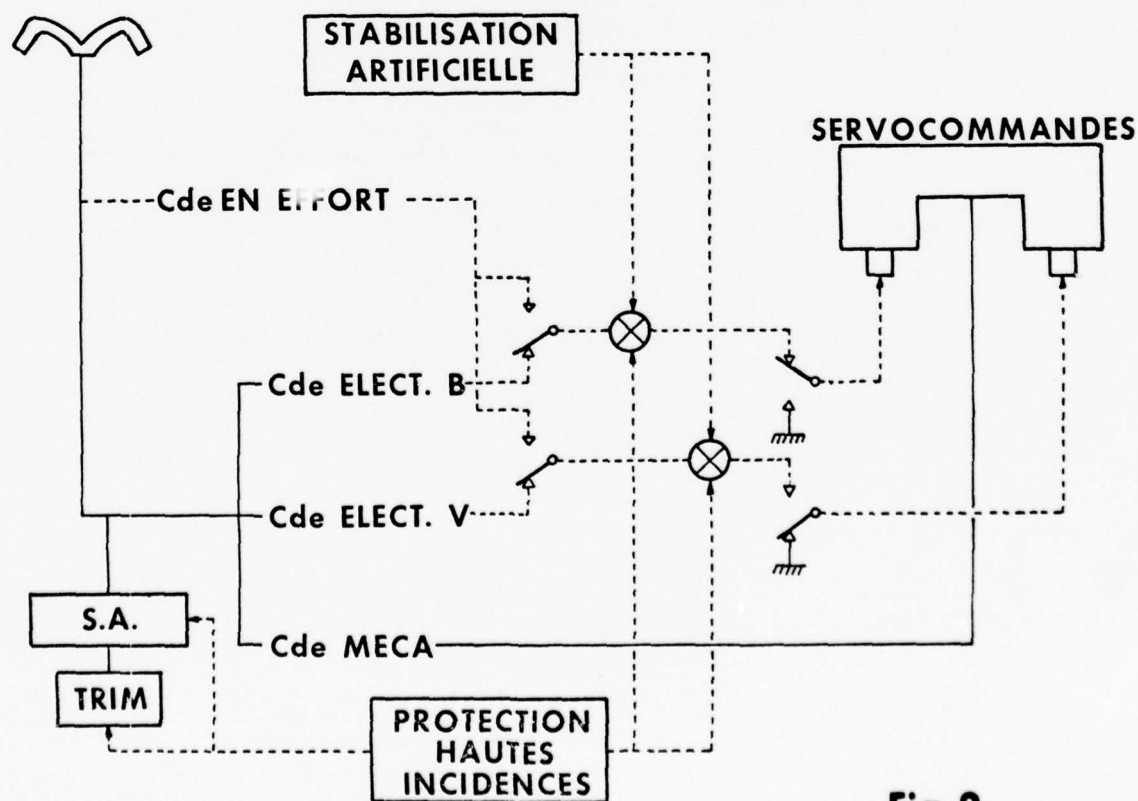
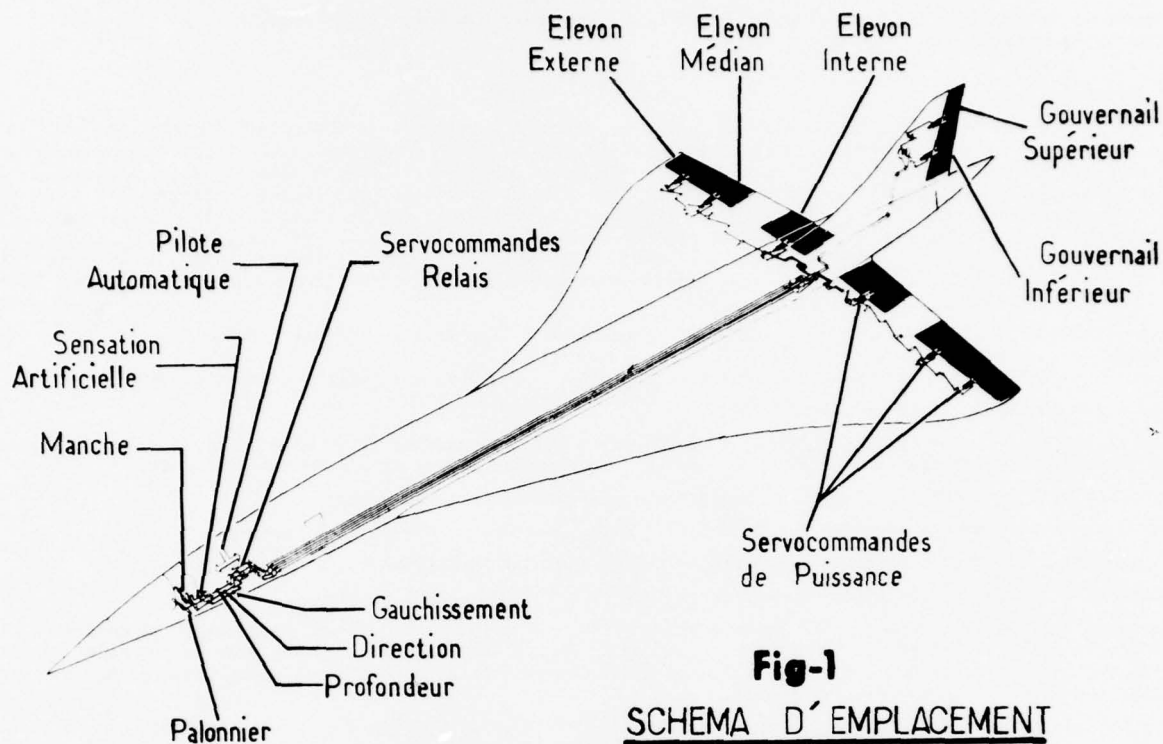
- la programmation est suffisamment aisée pour que les modifications inévitables lors du développement d'un avion puissent être réalisées aussi facilement (modifications simples) ou beaucoup plus facilement (modifications importantes) qu'avec un calculateur analogique.
- la programmation est suffisamment simple pour que des ingénieurs responsables de commandes de vol puissent la vérifier comme ils peuvent lire des schémas analogiques avec lesquels elle présente d'ailleurs une grande analogie.
- la rapidité de calcul permet tous les calculs de stabilisation de l'avion, même lorsqu'il convient de filtrer des réponses de modes structuraux.
- la fiabilité "au litre" est du même ordre que celle de bons calculateurs analogiques (ce qui représente un progrès notable puisque, au moins dans l'utilisation considérée ici, un litre de numérique accomplit nettement plus de travail qu'un litre d'analogique).

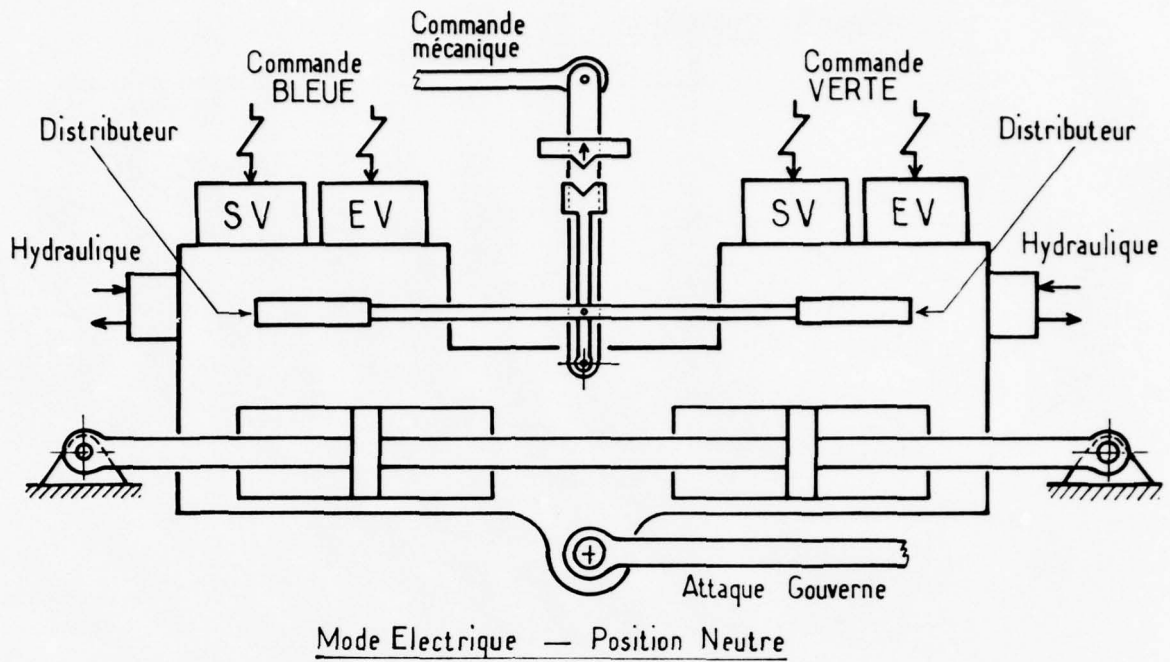
En aval des calculateurs, les commandes de vol ont besoin de servocommande. Il s'agit là d'une espèce comportant de nombreuses familles. Après la famille "simplex" de CONCORDE (et "triplex" mais à commande mécanique de l'AIRBUS), AEROSPATIALE a essayé dans ses laboratoires, un membre de la famille "quadruplex" dont les performances se sont révélées bonnes et même étonnamment bonnes en ce qui concerne la stabilité. Cette servocommande subit actuellement des essais d'endurance.

En amont des calculateurs, on se heurte non plus à de nombreuses familles, mais presque à une infinité : celle des organes de pilotage. Mettez ensemble quelques paramètres tels que position, effort, forme, mouvement contrôlés (tangage/roulis ou tangage/roulis/lacet), lois de pilotage associées, importance accordée au pilotage automatique ou transparent, morphologie des pilotes, possibilités technologiques (aussi !) et songez, pour vous rassurer, qu'une solution sera trouvée puisqu'il le faut. Et ainsi, puisque nous voulons essayer des lois de pilotage en vol, un minimanche a été réalisé et essayé au simulateur de vol. Bien entendu, il est très bon pour ceux qui en ont initialement choisi le type.

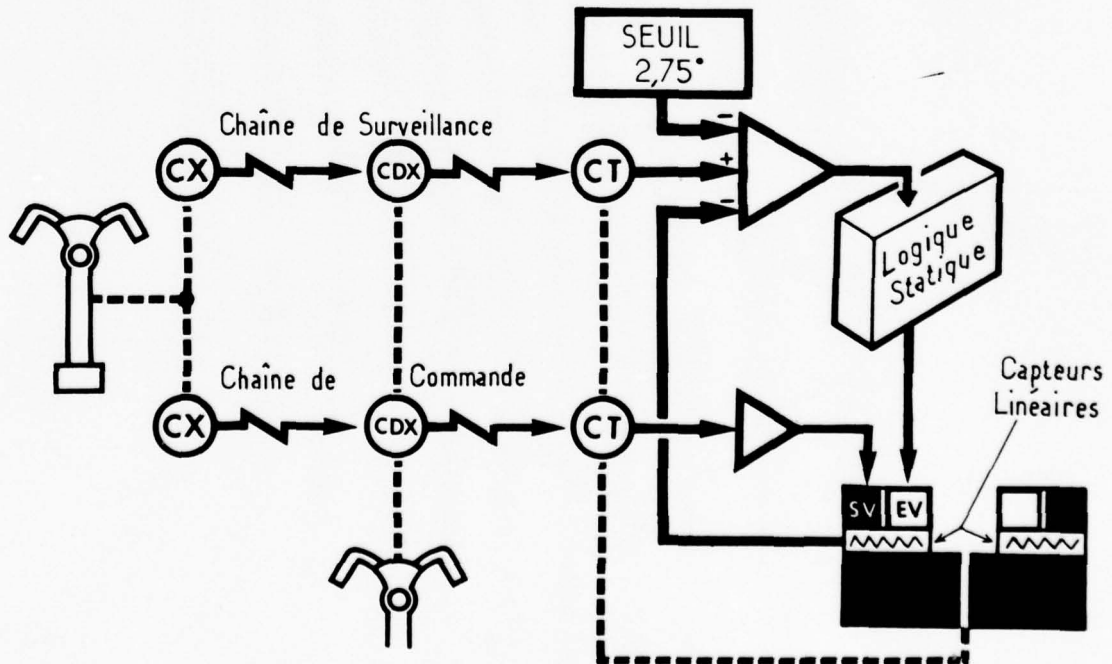
Il est prévu qu'un frère jumeau de ce minimanche sera installé dans quelques mois dans un avion CONCORDE pour expérimentation de lois de pilotage. Il sera accompagné de deux calculateurs numériques du type précédemment mentionné. Les essais permettront d'explorer un domaine de vol extérieur au domaine de vol de l'avion de série.

Par l'ensemble des travaux mentionnés ci-avant, AEROSPATIALE se prépare donc à l'utilisation de commandes de vol purement électriques. Elles ne constitueront d'ailleurs pas une révolution mais plutôt une évolution par rapport aux commandes de vol du CONCORDE type I de série dont, rappelons-le, la commande normale, électrique, a pleine autorité et assure intégralement sa propre surveillance.





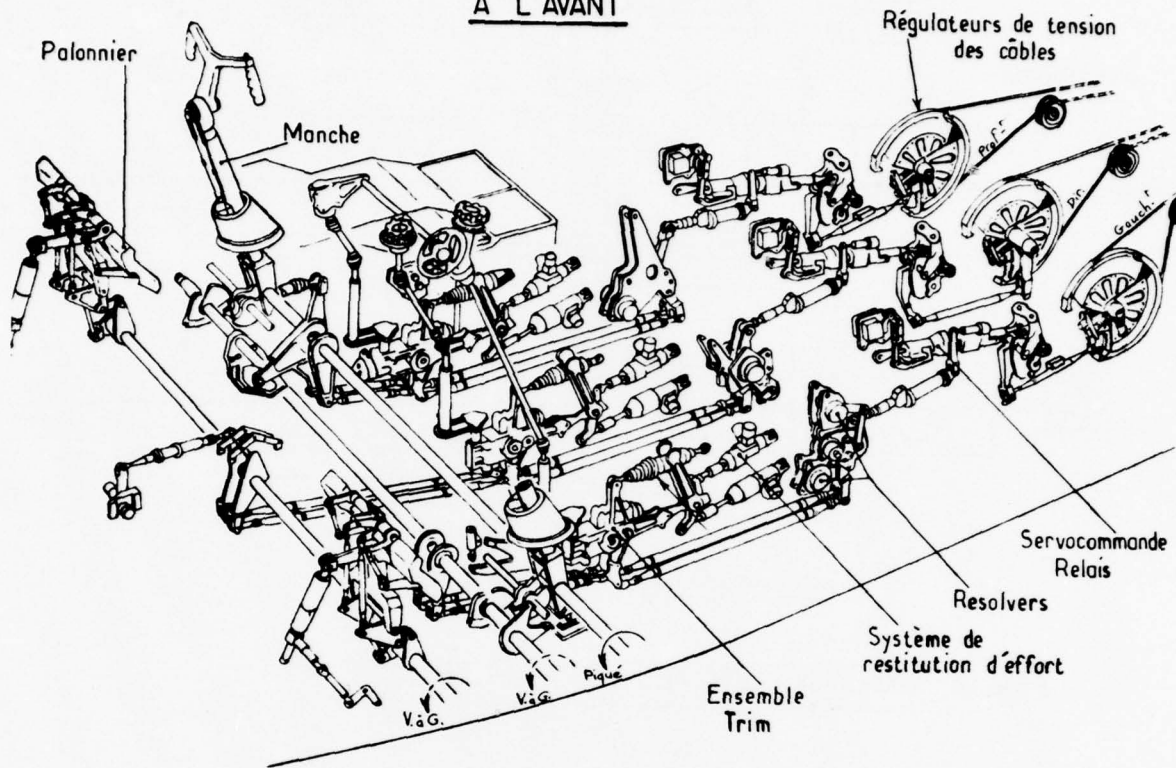
**Fig-3** SERVOCOMMANDE DE PUISSANCE



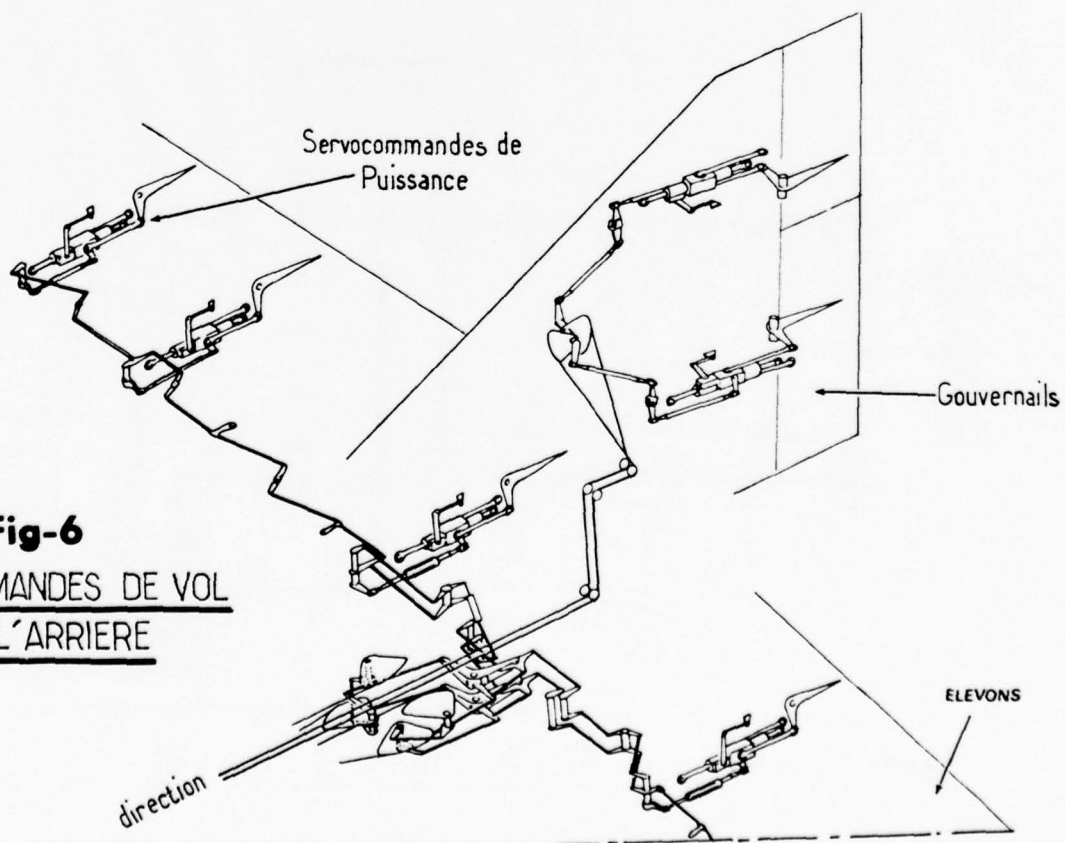
**Fig-4** CHAÎNE ELECTRIQUE BLEUE D'ELEVONS

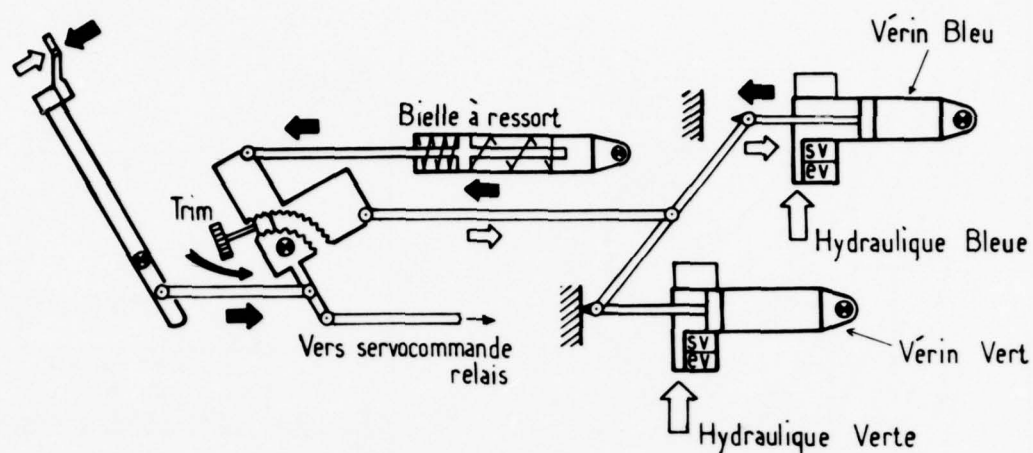


**Fig-5** COMMANDES DE VOL  
A L'AVANT

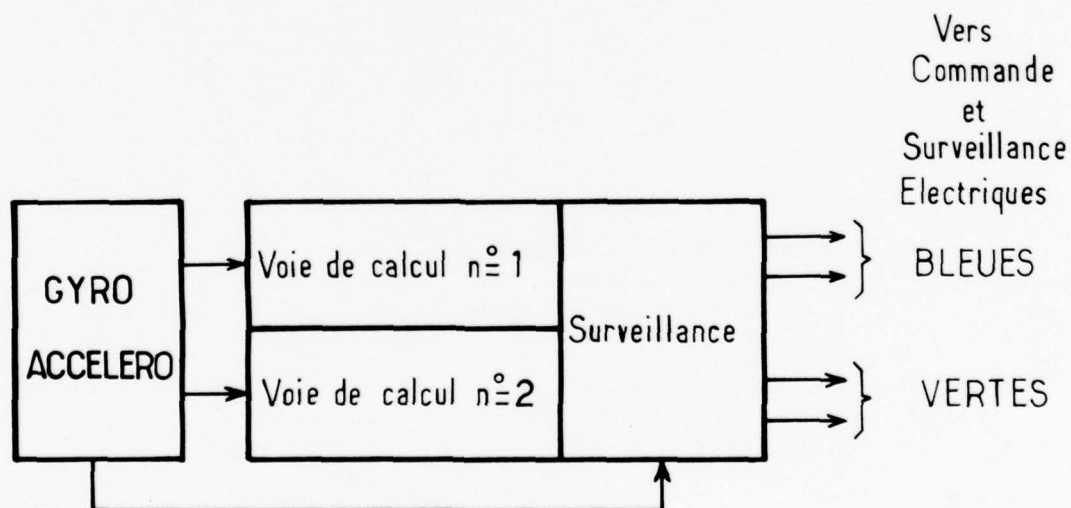


**Fig-6**  
COMMANDES DE VOL  
A L'ARRIERE



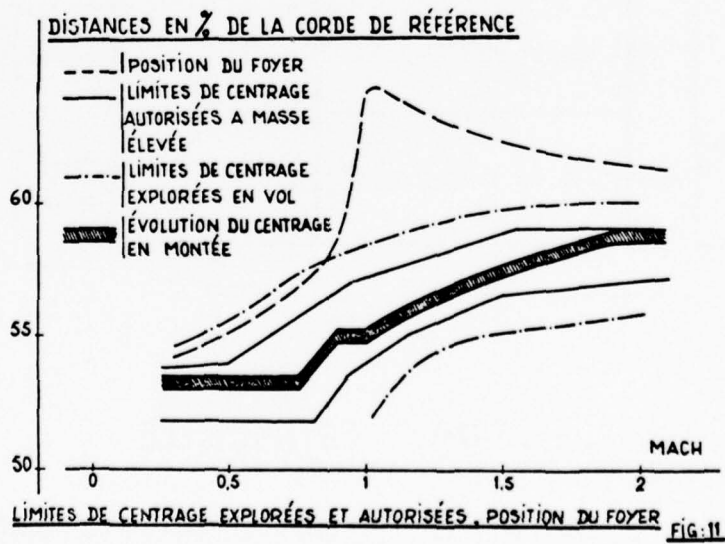
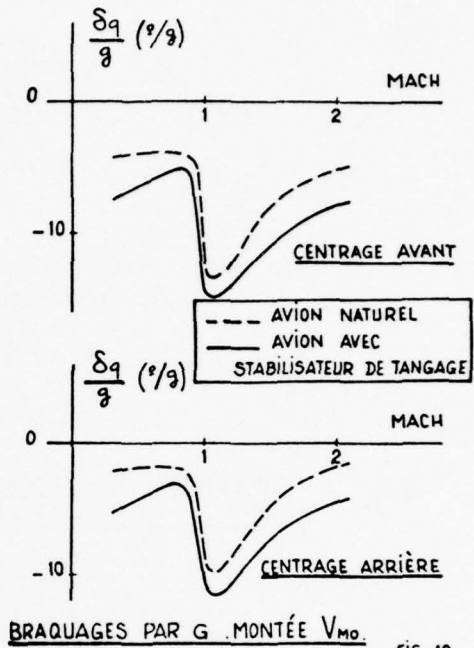
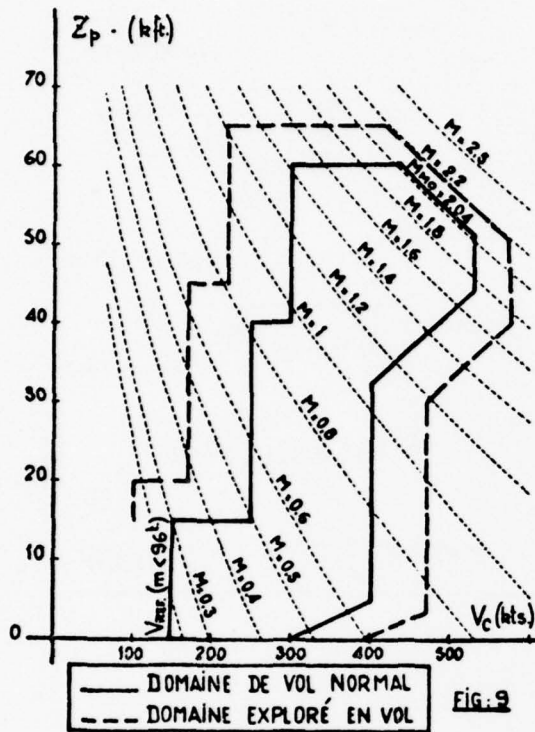


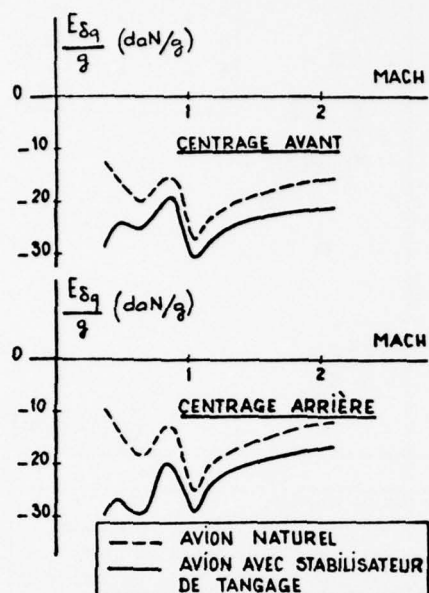
**Fig-7** SYSTEME DE SENSATION ARTIFICIELLE ET DE TRIM



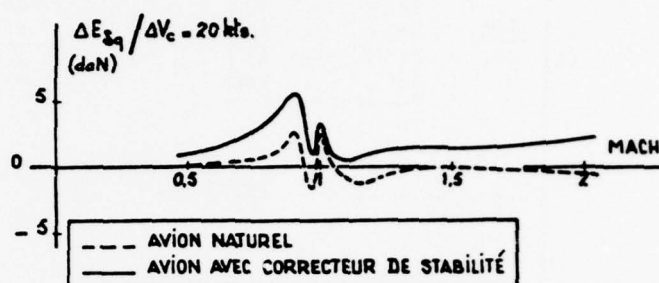
**Fig-8** STABILISATION ARTIFICIELLE

SCHEMA DE PRINCIPE D'UN DEMI-SYSTEME

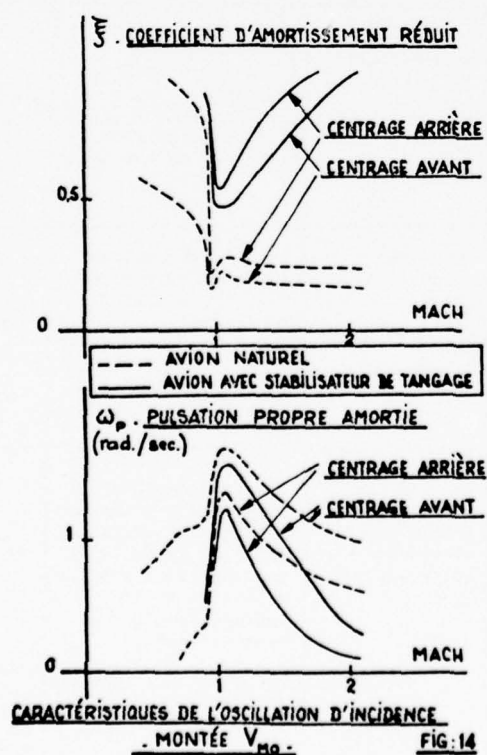




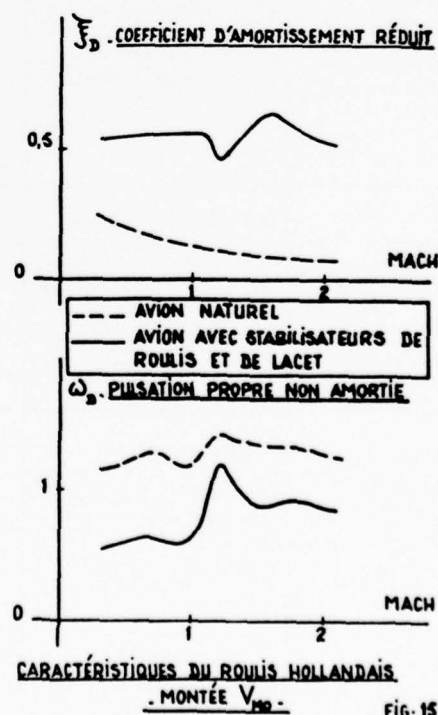
EFFORTS PAR G A LA PROFONDEUR AUTOUR  
DE  $N_3=1$  - MONTÉE  $V_{mo}$  FIG. 12



VARIATION DE L'EFFORT AU MANCHE POUR UNE VARIATION DE VITESSE  
DE  $\Delta V_c = 20$  kts. MONTÉE TYPE  $V_{mo}$  FIG. 13



CARACTÉRISTIQUES DE L'OSCILLATION D'INCIDENCE  
- MONTÉE  $V_{mo}$  - FIG. 14



CARACTÉRISTIQUES DU ROULIS HOLLANDAIS  
- MONTÉE  $V_{mo}$  - FIG. 15



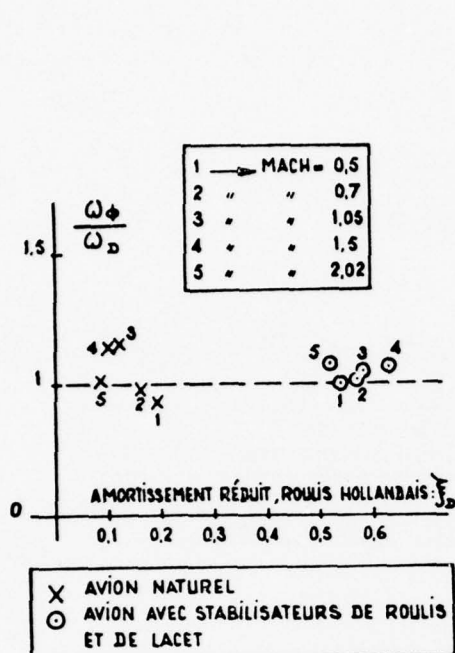
MONTÉE TYPE V<sub>MO</sub>

FIG: 16

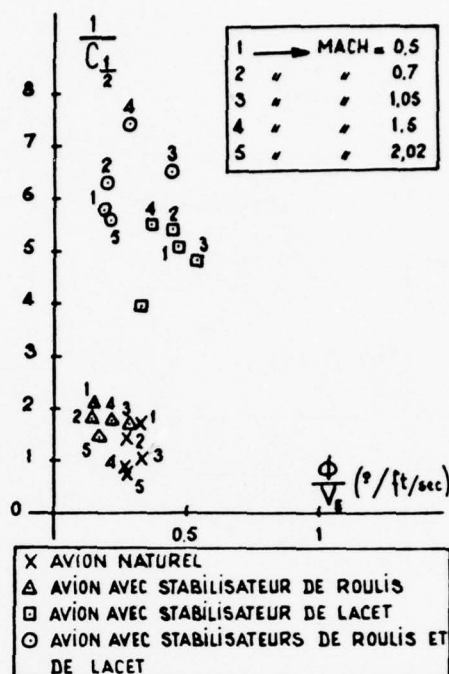
MONTÉE TYPE V<sub>MO</sub>

FIG: 17

## EXTRAIT DE L'ANALYSE DE SÉCURITÉ DES COMMANDES DE VOL

PANNES	CONSÉQUENCES	PROBABILITÉ PAR HEURE	PANNES	CONSÉQUENCES	PROBABILITÉ PAR HEURE
Perte de la commande électrique bleue des éleveurs internes seuls	Les éleveurs internes sont commandés par la chaîne de commande électrique verte. Aucune conséquence sur le pilotage.	$2.10^{-4}$	Perte de la commande électrique bleue + verte des éleveurs internes	Les éleveurs internes sont commandés par la chaîne mécanique. Dégradation du pilotage longitudinal (réduction de l'amortissement en tangage)	$8.8.10^{-7}$
Perte de la commande électrique bleue des éleveurs médians externes seuls	Les éleveurs médians et externes sont commandés par la chaîne de commande électrique verte. Aucune conséquence sur le pilotage.	$4.10^{-4}$	Perte de la commande électrique bleue + verte des éleveurs médians externes	Les éleveurs médians externes sont commandés par la chaîne mécanique. Perte de la stabilisation artificielle en roulis. Dégradation du pilotage, notamment en turbulence.	$1.6.10^{-6}$
Perte de la commande électrique bleue des gouvernes de direction	Les gouvernes de direction sont commandées par la chaîne de commande électrique verte. Aucune conséquence sur le pilotage.	$1.7.10^{-4}$	Perte de la commande électrique bleue + verte des gouvernes de direction	Les gouvernes de direction sont commandées par la chaîne mécanique. Perte de la stabilisation artificielle de lacet. Dégradation du pilotage latéral.	$7.7.10^{-7}$
Perte de la commande électrique bleue sur l'ensemble des gouvernes	Les gouvernes sont commandées par la chaîne de commande électrique verte. Aucune conséquence sur le pilotage	$7.3.10^{-5}$	Perte de la commande électrique bleue + verte sur l'ensemble des gouvernes.	Les gouvernes sont commandées par les chaînes mécaniques. La stabilisation artificielle est perdue sur les 3 axes. Le pilotage, bien que nettement dégradé, est possible en sécurité dans l'ensemble du domaine de vol autorisé.	$1.6.10^{-8}$

FIG 18

# A HIGH-RELIABILITY, HIGH-INTEGRITY FLIGHT CONTROL SYSTEM FOR HELICOPTERS

by

P. Robinson\*

J. Meadows†

C.M. Copage\*\*

\*Controls and Displays Division  
Flight Systems Department  
Royal Aircraft Establishment  
Farnborough, Hampshire, UK

†Head, Helicopter Systems Design  
Smiths Industries Limited,  
Aviation Division  
Cheltenham, Gloucestershire, UK

\*\*Head, Systems Safety Group  
Smiths Industries Limited  
Aviation Division,  
Cheltenham, Gloucestershire, UK

## SUMMARY

This chapter briefly describes some of the operations which helicopters may be required to carry out at night and in poor visibility. Because of the very high pilot work load likely to arise in these situations, it is argued that the helicopter should be equipped with an autostabilisation system having a defect-survival capability. One system to meet this requirement together with quantitative system reliability and integrity requirements, has been developed and manufactured by Smiths Industries Ltd for a Sea King helicopter at the Royal Aircraft Establishment, Farnborough; flight trials have just commenced. This system is triplex, with digital computation and has the development potential to include autopilot facilities, more sophisticated control techniques and extended system redundancy. The redundancy philosophy and the approach to assessment of system reliability and integrity are described, together with salient design and engineering details of the system. Also an indication is given of future trends in the technology.

## 1 INTRODUCTION

Helicopters, V/STOL and CTOL aircraft, especially the latter, have for many years been operating successfully at night. For many CTOL aircraft this has included poor visibility operation, however apart from specialised ASW vehicles, the majority of helicopters are still limited to good weather conditions and particularly good visibility. The reasons for this lack of progress are many.

What justifies the current activity in this field is largely the increased emphasis on operations which helicopters are now being required to carry out under IFR conditions. These are operations which not many years ago would not have been contemplated in conditions other than good, day-time visibility. There is now a real possibility that with the help of developments in electronic vision aids, limited visibility operations can be attempted.

It is the aim of this paper to postulate some implications on the flight control system of operating helicopters at night and in poor visibility. The principal requirements for the flight control system, to alleviate the piloting problems in these conditions will be discussed and the second part of the paper describes one particular solution to these requirements, namely the use of a triplex, digital autostabiliser currently under flight trials in a Sea King helicopter.

## 2 HELICOPTER OPERATIONS IN POOR VISIBILITY

In the UK, it is probably true to say that the Army Air Corps have the most pressing requirements to extend their operations at night time and in poor visibility. Army helicopters are employed in many roles, most of which require extremely low-level flying and agility of manoeuvre at some stage of the mission. It is these two features which will undoubtedly give rise to problems in conditions other than clear daylight.

Up to the present time, the Army (light/medium) helicopters have not been equipped with autostabilisation. The introduction into service of the WGL3 'Lynx' marks the change, with the comprehensive, duplex AFCS, fitted to all variants.

The Royal Navy on the other hand, has for many years been successfully operating helicopters over the sea at night and in poor visibility. Wasp, Wessex and Sea King helicopters are all equipped with an AFCS. For the ASW role the Wessex and Sea King equipment includes the facility to descend, transition and hover automatically at very low heights. With the exception of the duplex system in the Wessex Mk.3 (and the Lynx, when in service) a simplex, limited authority AFCS has always been fitted.

The AFCS fitted to Royal Marine Commando helicopters has not differed from that fitted to the corresponding Royal Navy aircraft. The most demanding Commando role in terms of night/poor visibility operation is the amphibious assault, since once again it requires very low flying over both sea and land.

It is apparent that many of the problems associated with operation at night and in poor visibility arise from the helicopter's low inherent stability which requires continuous monitoring of vehicle attitude together with frequent control action, especially in turbulent conditions. In clear weather, much of the required attitude information is derived from external references and this action is easily integrated with other visual functions both inside and outside the cockpit. However, even in 'limited' instrument flight (e.g. cruising at altitudes well above obstacle heights) the tasks of continuously monitoring and maintaining attitude from an artificial horizon, monitoring other flight, engine and systems instruments, operating radios and navigating, present a very full work load. If in addition, operations are to be extended to encompass some of the previous low-level roles, for which it is essential for the pilot to 'see' the ground ahead, then the problem is even more acute.

It is likely that the first viable means of providing the required forward vision will be a LLL TV sensor with a head-down CRT presentation to the pilot. In fact, with the exception of night goggles or other helmet-mounted devices, head-down CRT presentation of outside world information could be the common feature of the various sensing system currently under development. It is self-evident that none of these displays can match the total capabilities of the human eye in good visibility and in particular, a compromise must always be sought between resolution and field of view. Thus in addition to the work load briefly described already, must be added the task of closely monitoring the CRT in order to interpret ground features, detect obstacles (power cables in particular) and targets, judge clearance heights and so on. In fact this task alone could occupy a major portion of the pilot's time since the quality of the CRT image will be such that close and continuous scrutiny will be required as opposed to the clear visibility situation where a quick scan augmented by peripheral vision capability is adequate.

In order to reduce the work load to a tolerable level, the benefits of automatic stabilisation must be exploited. By providing autostabilisation in pitch, roll and yaw, the pilot is relieved of the task of making continuous control adjustments to correct for short term disturbances, the handling is improved and a 'hands-off' capability is acquired when flight conditions allow this. Further gains can be made by the addition of autopilot modes to the basic stabiliser but the selection of the appropriate modes requires careful consideration in the strict context of the roles previously described since the simpler modes (such as altitude hold and acquire) may be of limited use in those regimes where the work load is highest. Approach modes and transition/hover modes (for Naval applications) constitute notable exceptions however.

The maximum reduction in work load can only be achieved if the pilot has considerable confidence in the autostabiliser and its integrity. Several factors affect the confidence.

- (i) the behaviour of the system and aircraft following an autostabiliser failure. This can be separated into the immediate consequence, i.e. the severity of the transient disturbance in height, speed and attitude and then the ensuing degradation due to reversion to the unstabilised helicopter,
- (ii) the probability of a failure occurring during the mission,
- (iii) the existence or otherwise of a warning indication of the failure occurrence.

The important question of integrity is discussed later but in the present context it must be stated that a viable autostabiliser for night and poor visibility operations must meet the requirements implied in the above factors.

### 3 AFCS REQUIREMENTS FOR NIGHT/POOR VISIBILITY OPERATION

In this section, some requirements of an AFCS, relevant to poor visibility operation of helicopters in the preceding roles, are discussed briefly.

#### 3.1 Handling

Since this is the justification for incurring the cost/weight penalties of an AFCS, the benefits derived must be substantial. Recent trends and future requirements in helicopter design and operations (such as poor visibility operation, long sorties, single pilot operation, higher speeds and increased agility) depend on the enhanced flying qualities derived from autostabilisation. Satisfactory handling is achieved by the use of relatively simple and well established control techniques of attitude and rate feedback. There has been little requirement and consequently little effort (compared to fixed wing) to develop more sophisticated control techniques. However, trends towards increased agility and the increasing use of the helicopter as a weapons platform will transform this situation.

#### 3.2 Integrity

Until more experience is gained in the night/poor visibility operation of helicopters in roles such as those described previously, it will not be possible to say how dependent the mission will be on the functioning of the AFCS. It is suggested that many missions could still be completed in the event of an AFCS failure, albeit with greatly increased work load and degraded performance. Unfortunately, the performance degradation particularly as measured against the requirement to avoid detection may well decrease the probability of surviving the mission.

What are crucial, however, are the failure transient characteristics of the system, particularly in the low-flying situation. If the pilot is aware that a single defect in the system can result in rapid and dangerous changes in attitude and height unless he intervenes immediately, then the confidence that is so necessary for the full benefits of the system to be realised will be lost.

An AFCS runaway in a simplex system would almost certainly be unacceptable, even with a modest AFCS authority, therefore some degree of redundancy is essential. Although a duplex, fail-soft configuration offers substantial improvements over a simplex system there is a strong case in favour of a three-lane system. The duplex system is capable of providing a failure warning indication and produces tolerable



failure transients in many situations but it must be left to the pilot to decide on and isolate the faulty lane, possibly in a high work load situation. A three-lane system on the other hand can be designed with negligible failure transients and both detection and isolation of the first defect or its suppression is rapid and automatic. The term "three-lane" is used rather than "triplex" to permit the latter to stand for the type of parallel operation in which a decision that a lane ceases to contribute to the system output is based, substantially if not wholly, on the relative behaviour of all three lanes.

### 3.3 Authority

Adequate stabilisation is achieved on current helicopters with modest authorities of the order of 10%. When radically different vehicle designs are contemplated and the requirement arises for revised control techniques (e.g. manoeuvre demand), authorities of this magnitude will no longer suffice. Higher authority would of course demand consideration of the overall system characteristics including power and hydraulic supplies.

### 3.4 Test and maintenance

One of the arguments against additional AFCS complexity implied by redundancy of equipment is that it necessarily decreases the overall reliability of the system in terms of MTBD and consequently increases the maintenance man hours per flying hour. In addition, multilane systems can suffer from the problem of 'nuisance disconnects' arising from inevitable equipment tolerances. These can give rise to reduced system redundancy, with no defect revealed on investigation.

However, there is much that can be done to minimise the problems. The inclusion of built-in-test equipment can provide a comprehensive pre-flight check with a GO/NO-GO indication of system functionality and redundancy. Likewise, a continuous in-flight monitoring can provide a high degree of both fault detection and location down to LRU level. Finally, the same facility can provide a first-line test capability without the need for additional test equipment (in the multilane configuration).

By suitable design, the occurrence of nuisance disconnects can be considerably reduced, especially if the system is implemented digitally, in which case built-in-test can be incorporated with fewer additional components than in an analogue system. Also, for given component tolerances, the comparison of like-with-like in a multiplex configuration is less susceptible to nuisance disconnects than the self-monitoring in a multiplicate configuration.

### 3.5 Cost and weight

There is little of a general nature that can be said about this, other than the obvious desire to minimise both. Rarely is something gained for nothing and it must be accepted that if a high reliability, high integrity AFCS can substantially increase the feasibility of safe night/poor visibility operation then the increase in cost and weight of the system is the price to be paid.

## 4 THE RAE (FLIGHT SYSTEMS) SEA KING PROGRAMME

### 4.1 General

A Sea King Mk. 1 helicopter is currently being used at RAE for trials of helicopter avionic systems and is being equipped with a range of displays, controls, navigation and other equipment. The central theme of the programme is the need to develop the various equipments to meet the total requirement for night/poor visibility operation. In addition it can be used to evaluate equipment and procedures for helicopters being introduced into service and to investigate ad hoc problems of in-service helicopters.

### 4.2 Controls programme

The long term AFCS requirements of UK military helicopters are under continuous review, in order that the work centred on the Sea King can be aimed to meet them. Two areas in which future development is seen to be most likely are the introduction of a greater degree of system redundancy and the trend towards digital implementation of systems.

With this in mind, a specification was issued for a defect-survival autostabiliser (DSAS) to be developed for installation and flight trials in the RAE Sea King. It was made clear that a digital solution would be preferred in order to encourage development in this area and to gain some early flight experience of digital techniques applied to flight control systems.

At the same time as meeting these research objectives it was felt that defect-survival stabilisation was desirable in the RAE Sea King for carrying out other areas of the overall programme. The contract was awarded to the Aviation Division of Smiths Industries Ltd., the equipment was delivered at the end of 1975, and flight trials have just commenced.

### 4.3 Particular requirements of Sea King DSAS

The performance requirements for the DSAS in terms of attitude holding, stability, response to pilot demand etc. are such that they can be met in a conventional manner using attitude and rate feedback. No new performance criteria involving additional sensors and more sophisticated control philosophies were required for the initial part of the trials, but will be considered subsequently.

In terms of reliability of the multiplex mode, it is specified that any single defect of the system shall have negligible effect on the autostabilisation performance. In particular, the maximum transient deviations following a single defect in any axis during trimmed, 'hands-off' flight should not differ from the nominal, trimmed values by more than the following amounts:-



	Attitude (deg)	Rate (deg/s)
Pitch	1.0	1.0
Roll	1.0	1.0
Heading	2.0	1.0

The pilot should be informed of the occurrence of the first defect and should be able to assume manual control of the aircraft in the event of a system failure, allowing for an intervention time of 1.5 s. Other aspects of the integrity of the multiplex mode are that the total risk of multiple defects, which result in deviation in two or three axes from the nominal, trimmed values exceeding the following amounts, should be less than  $10^{-7}$  per flying hours:-

	Attitude (deg)	Rate (deg/s)
Pitch	2.0	1.0
Roll	2.0	1.0
Heading	2.0	1.0

The reliability of the AFCS currently fitted in the Sea King is good, and one of the design aims for the DSAS is to achieve similar standards of reliability whilst at the same time effecting significant improvements in mission success by ensuring a lower probability of losing the autostabilisation facility; specifically an MTBD in excess of 400 flying hours and an MTBF of more than  $10^5$  flying hours in the multiplex mode. The test facilities outlined in section 3.4 form part of the DSAS specification.

In order to maintain as short a development timescale as possible, so as to acquire early flight experience, the original requirement to carry the defect survival philosophy into the actuation area has been postponed. Therefore the DSAS is multiplexed to the extent of sensing and computation only. The interface with the existing simplex actuators and power controls in the Sea King will be described in the second part of the paper. Finally, it was required that it should be possible to integrate the new autostabilisation system with the autopilot facilities of the existing AFCS.

#### 4.4 DSAS flight trials

After installation of the system in the Sea King, ground checks will be followed by preliminary familiarisation flights over the flight envelope which will include an assessment of the failure transients in order to clear the system for further work. The performance of the system, as a stabiliser, will be evaluated in terms of stability, attitude holding and handling over a range of turbulence levels and flight conditions.

A detailed study will be made of the system behaviour following any genuine defect and also specific, induced, 'defects' that will be selected to appear in critical areas of the system. Limited access to data within the computers will be available for recording purposes, a particular interest being sensor signals before and after consolidation. Thus the consolidation process can be assessed and the comparator settings inherent in this process can be confirmed. A range of comparator values can be selected in order to investigate nuisance disconnect problems.

Because the equipment is digital, its immunity to electrical noise in the aircraft environment will require examination, as will the actuator response to quantised inputs (a separate rig exercise using Sea King actuators checked this latter point, during the development of the system).

One feature that it will not be possible to check in flight to a significant level of confidence is the overall system reliability. This is unfortunate but is due to the relatively small number of flying hours that can be achieved in any experimental aircraft.

### 5 DIGITAL COMPUTER DEVELOPMENT

The Aviation Division of Smiths Industries Ltd., has been developing special purpose digital computers for airborne applications over the last decade, including those for the Jaguar, Harrier and MRCA Head-Up display systems, a duplex digital installation for an engine control system and, more recently, computers for a triplex AFCS providing all of the autopilot modes required by present generation transport aircraft, including automatic landing in Category III conditions. At the present time development of small processors using LSI technology is proceeding within the Company.

The ground rig evaluation of the triplex AFCS system mentioned above was successfully completed in 1973, and this digital computer, the SDC 10, was selected for further development for the Sea King system.

### 6 THE REDUNDANCY PHILOSOPHY

6.1 Because digital computing techniques are to be used, self-monitoring lanes can be considered. In particular two such lanes could be used in a parallel-duplicate configuration, where this stands for the type of parallel operation in which a decision that a lane ceases to contribute to the system output is based, substantially if not wholly, on its own behaviour. However, the level of self-monitoring required is so high, e.g. 99.94% for a lane MTBD of 1200 flying hours, that even its achievement is questionable, let alone the demonstration by failure analysis of such achievability. With currently available equipment, it is considered that the triplex configuration selected for this installation represents the minimum level

of redundancy providing a defect-survival system capable of achieving the mission-success probability, whilst also providing a fail-soft characteristic following a second defect.

If giving a warning following a single defect had not been a requirement, then a Parallel-triplicate configuration could have been considered. The level of self-monitoring required is now practicable, e.g. 93% for a lane MTBD of 1200 flying hours. However the work involved in failure analysis would be much greater than for a triplex configuration; also the parallel-triplicate configuration would provide fail-operational or fail-soft behaviour on only 89% of occurrences of two defects which might well be as unacceptable as a warning on only 93% of occurrences of a single defect.

Dependence on in-lane monitoring would require each sensor to be more complex than for triplex, in that it is either self-monitored or interrogatable by its computer in flight. This could be avoided of course by cross-comparing and amalgamating sensor signals as for triplex, but one would then sacrifice the inherent advantage of a parallel-triplicate configuration; i.e. obvious independence of lanes except at the final consolidation. The latter could comprise fan-in of all three lanes to each actuator coil, i.e. double fan-in to each actuator, in this application and quite possibly would require long-term datum equalisation to prevent build up of large sustained differences due to sensor tolerances and 'integration' in the control law. In comparison, each actuator coil is wholly in-lane for the triplex configuration.

6.2 The triplex configuration operates upon the principle of a two out of three majority vote, whereby a faulty lane is identified and subsequently rejected as a result of differences in some specific performance index or indices, and the corresponding performance indices of the two good lanes. For failure detection, such a system must obviously allow the build-up of some level of lane differences prior to initiating a lane rejection; consequently it is necessary, as with all monitored systems, to find a compromise solution which on the one hand keeps the aircraft transient excursions following a defect and lane rejection within the specified levels, and on the other hand does not result in an unacceptably high incidence of nuisance lane rejections. From this point of view the use of digital processors in the multiplex configuration is particularly advantageous since tolerance effects normally associated with the computing functions in analogue systems can be completely eliminated. However, the failure analysis, see section 13, pessimistically, assumed a nuisance-disconnect rate for the digital processor equal to its defect rate.

Integration with the existing Sea King power supplies and actuation system has prevented the extension of the defect survival philosophy in these areas during the present programme; however, such extensions are seen as logical developments for future systems.

The triplex philosophy has previously been successfully applied by Smiths Industries for the Trident AFCS. This was the first (and to date, i.e. mid-1976, the only) aircraft to be certificated for Category III operations by the CAA and this background of experience is now supported by the successful implementation and testing of the triplex digital control system previously discussed, the main features of which are being incorporated into the DSAS.

## 7 CONTROL TECHNIQUES

The system achieves autostabilisation of the helicopter in a conventional manner, namely pitch and roll stabilisation by operating respectively the fore and aft, and the lateral cyclic control of the main rotor in response to rate and attitude error information, and yaw stabilisation by operating the tail rotor blade angle collectively in response to yaw rate signals. A limited heading hold facility is also provided by operating the tail rotor in response to proportional and integral of heading error signals. This latter facility is automatically cut out when the pilot pushes on the yaw pedals.

The attitude and rate information required for the pitch and roll axes may be obtained in a number of alternative ways:

- (i) by using separate rate and attitude sensors
- (ii) by using a single sensor to measure attitude and deriving a rate signal component by phase advance techniques,
- (iii) by using a single sensor to measure the rate and deriving an 'attitude' component by integration or phase lag techniques.

Helicopter systems so far designed and developed have normally been based upon either (ii) or (iii) and for Naval applications, involving considerable periods of low speed or hovering flight at low altitude, the attitude based systems have generally been preferred, since such systems can provide a stable long term attitude datum. Since the rate gyroscope is basically a simpler, and consequently a more reliable device than the vertical gyroscope, it was decided to use the very high reliability Series 700 gas bearing rate gyroscopes manufactured by Smiths Industries Limited as the primary system sensors, providing both rate and short term attitude signals, and to obtain long term attitude signals from the three vertical gyroscopes fitted as basic equipment in the RAE Sea King helicopter.

The specified short term attitude stabilisation performance can be achieved in the absence of any vertical gyro signals. However, under these conditions there is some deterioration in the long term attitude hold performance. During turning flight the pitch and yaw rate signals are combined as a function of roll attitude to mitigate the loss of height which would otherwise occur as a consequence of using body-fixed rate gyroscopes.

---

\* One of the assumptions involved in these estimates is that a defective lane which does not cut itself out would have equal probabilities of 'positive' spurious activity, spuriously-zero activity and 'negative' spurious activity.

The autostabilisation system operates through series actuators, having an authority limitation of  $\pm 10\%$  of full control travel in pitch and roll and  $\pm 5\%$  in yaw. To avoid saturation of this limited authority during manoeuvres involving high rates and/or large attitude changes, electrical command signals proportional to the fore and aft and the lateral displacements of the cyclic control column are generated by position pick-offs and combined with the rate gyro feedback terms.

Trimming signals are also injected into the system by means of the pitch, roll and heading trim knobs on the pilot's control panel. A hover indicator provides the pilot with measures of the pitch, roll, yaw and collective actuator deviations from their central positions. The defective survival autostabilisation system is integrated with the standard hover indicator on the interseat console.

An initial assessment of the system performance and control law optimisation was carried out using a general purpose hybrid computing facility. These studies included an examination of the effects of variations in the computer iteration rate, and of quantisation of the A/D and D/A converters. From these investigations it was decided to incorporate 12 bit A/D and 10 bit D/A converters and to use a 55 millisecond computer cycle time.

## 8 FEATURES OF THE REDUNDANCY CONFIGURATION

### 8.1 The DSAS system comprises the following units:

- three SDC 10 digital computers
- a triplex rate gyroscope for each axis of control, i.e. pitch, roll and yaw
- a triplex cyclic control column position pick-off for both pitch and roll control,
- a pilot's control unit incorporating triplex pitch and roll trimming facilities and a simplex yaw trimmer.

The system interfaces with the following existing facilities in the RAE Sea King helicopter:

- three identical vertical gyroscope generating three wire synchro pitch and roll attitude signals,
- the compass system providing a three wire synchro heading signal,
- the autopilot pitch and roll command signals from the Mk. 31 AFCS,
- yaw pedal discriminants to cancel the heading hold facility,
- the series connected pilot and co-pilot cut-out buttons,
- the interseat null indicator

A schematic of the triplex configuration and its integration with the existing actuation system is shown in Fig. 1.

### 8.2 Sensor Inputs

The triplex rate, stick pick-off and trimmer signals are fed on a lane one sensor to lane one computer basis. Following A/D conversion, the sensor data is serially crossfed between computers using high integrity interlane data highways and subsequently amalgamated within each computer to form the mean of three in the triplex mode, or the mean of two in a duplex mode. This amalgamation process ensures that all three digital computers are processing identical input information and consequently generating identical digital output data. The attitude information from the three vertical gyroscopes used for long term pitch and roll control is also fed on a sensor one to computer one basis. In this case however, since the pilot will normally be monitoring his attitude information from the No. 1 vertical gyro by means of his attitude indicator, the signals from this unit will normally be used for control computations in all three lanes. The signals from all three gyros will be compared for monitoring purposes and in the event of a failure in the No. 1 gyro unit, automatic switch over to No. 2 vertical gyro will occur. In the event of this gyro also failing, the long term attitude monitoring capability will automatically cut out.

### 8.3 Actuator command signals

In the existing Sea King AFCS, the simplex autostabilisation input signals to each auxiliary actuator are fed as a common parallel input into two servo valve coils. In order to provide a measure of failure survival capability with the existing actuation system it was decided to input each coil separately from No. 1 and No. 2 computers respectively, the third computer output being fed back to its own A/D converter through a simple resistive model of the coil.

With this configuration two reversionary conditions may exist following the first defect, namely,

- (i) a duplex system if computer 3 fails, or
- (ii) a duplex system with one active output if either computer 1 or 2 fails.

In the latter condition, the overall system gain will be maintained by doubling the gearing in the working processors. The pilot can attempt to reinstate the triplex situation by momentarily pushing the POWER switch on the PCU into the forward ENGAGE position.



Following total system rejection due to a second defect, the pilot has the option of attempting to operate in a simplex mode on either lane 1 or lane 2 by using a CHANNEL SELECT switch on the pilot's control unit and re-engaging the system. For safety reasons it is arranged that it will be impossible for the pilot to engage the system in a simplex mode directly from the POWER OFF condition, and it is not possible to revert from simplex to multiplex operation without removing power and re-engaging the system.

#### 8.4 Computer synchronisation

The three digital computers operate at their own independent clock frequencies and the accumulation of excessive drift between them is prevented by software synchronisation techniques using synchronisation status pulses transmitted between computers along dedicated sync-stat highways.

The synchronisation routine is divided into two sub-routines, a 'coarse' synchronisation which pulls the computers to within a few instructions of one another following start up, and a 'fine' synchronisation routine which pulls all computers into line. A synchronisation window is defined in terms of programme steps, such that if any one computer falls outside this window, the amber warning in the pilot's control unit is operated, indicating a non viable triplex system where loss of a single computer due to a further defect could result in a system cut-out.

#### 8.5 Engage-disconnect facilities

The engage-disconnect system has been designed according to the principle that a faulty computer is not required to make any decisions with respect to either its own or any other computer's validity. In the triplex mode of operation a faulty computer is rejected as a result of validity assessments made by the other two computers. Disconnection of the analogue output signals is achieved by means of duplicated logic circuits in each computer which contain a pair of relay contacts, controlled by validity discriminants transmitted from the other two computers via dedicated interlane transmission lines.

For the duplex or monitored simplex modes of operation, measures are required to ensure that in the event of a computer failing such that it can no longer control the disconnect relay in the other computer, the latter can nevertheless still be cut out. This is achieved by means of a second pair of relay contacts in the engage-disconnect logic circuits which are controlled by two self generated validity discriminants.

The pilot is provided with the option of flying in a simplex mode following a second defect by using either lane 1 or lane 2. This is achieved by moving the CHANNEL SELECT toggle switch on the pilot's control unit into the appropriate position where switch contacts will bypass the engage-disconnect logic in the selected lane and ensure that there will be no output from the other lanes.

#### 8.6 Warnings

The multiplexed sensor signals are monitored for excessive differences, in which event, the faulty sensor signal will be excluded from the amalgamation process, the system continuing to operate using the mean of the remaining two input signals. The AMBER integrity warning indicator (Fig. 2) will be lit and the appropriate LRU symbol displayed on a seven bar display on the front of each computer.

The amber integrity warning light will also be operated as a result of a lack of synchronisation between computers and in the event of any computer power line relay being open circuit.

When a complete system cut-out occurs, a RED integrity indicator on the PCU is illuminated.

### 9 INTERLANE DATA TRANSMISSION SYSTEM

Inter-processor transmission links are used for the serial transmission of digital data between the three computers. This inter-change of data is required:

- (1) to consolidate the individual lane sensor information, thereby providing common input data for all three computers in order to derive the maximum benefits bestowed by the identical processing capabilities of the three computers,
- (2) to formulate difference signals for the identification of faulty sensors or computers and to generate warning and cut-out discriminants when the computer parameters exceed pre-defined levels.

The design of a high integrity interlane data transmission system is essential in order to preserve the failure survival capability conferred by the triplex redundancy. The transmitter-receiver system must be such that there is no possibility of misinterpretation of transmitted data in one receiving computer and correct interpretation in the other, due for example, to weak transmission from the sending computer, since such a situation may result in an initial rejection of a perfectly good computer subsequently leading to a complete system cut out. Measures to avoid this type of situation have been implemented in the transmission system and the associated software.

The practicability of replacing direct electrical interconnections between computers by the use of fibre optic transmission links has already been demonstrated by Smiths Industries during the experimental investigation of a duplex engine control system. For the Sea King it was decided to retain electrical interconnections in the interests of both development time and project costs. In any case, fully developed optical links do not automatically supersede electrical links because the latter can certainly be designed to meet current EMI Control Plans; i.e. a choice would be based on cost and reliability comparisons.



## 10 PRE-FLIGHT AND IN-FLIGHT TEST FACILITIES

### 10.1 Pre-flight testing

The pre-flight tests, which are initiated by one actuation of the test button on the pilot's control unit, form an essential process in the achievement of the desired integrity, ensuring a fully operational system prior to take-off. To avoid inadvertent in-flight operation of the test button, this is covered by a spring loaded hinged flap. The face of the test button is divided into an upper white panel displaying the legend TESTING when illuminated and the lower red and green panels displaying NO and GO respectively when illuminated. The tests are also terminated by pressing the button.

The pre-flight tests have been integrated with the pilot's normal pre-flight drill and require pilot participation when the white TESTING segment of the button is flashing. Prior to these an automatic test sequence is entered, and after them the green GO panel is illuminated if no fault is detected, or the red NO indicator is lit and the appropriate faulty unit is indicated on the computer seven bar displays.

In addition to the rate gyroscopes tests discussed in section 12.3 the following are some of the basic features which are checked out during the tests:

- (a) Correct functioning of all indicator lamps.
- (b) Ability to effect engagement in the multiplex mode, the full in-flight program being exercised using 'fixed' sensor data.
- (c) Tracking accuracies and scale factor of:
  - (i) pitch and roll stick pick-offs
  - (ii) pitch and roll trimmer output signals.
- (d) Heading trimmer sensitivity and heading disconnect logic.
- (e) Correct functioning of pilot and co-pilot autostabiliser cut-out buttons.
- (f) Engage and disengage logic.
- (g) Correct functioning of A/D and D/A converters.

Additional software checks on the correct functioning of the computers are also carried out, together with the in-flight tests discussed below.

### 10.2 In-flight testing

In addition to those tests required for sensor and output data monitoring purposes, performed each computing cycle as part of the basic in-flight operating programme, a number of additional in-flight checks are carried out to provide added confidence in the correct functioning of the computers when operating in the airborne environment. These include store checksums, checks on the engage-disconnect programmes and limiter sub-routines.

### 10.3 Installation testing

In the preflight check the program halts only when the pilot's normal preflight drill is involved and at conclusion of testing, so that the GO indication has to be acknowledged. However, system status indications are flashed through in the automatic test sequence and the program has been provided with a 'halts' option covering each change of the Amber Integrity, Red Integrity, Lane In/Out, or Null flag indication. This option should be taken up after any disturbance of the system installation so that the inspector signing for the autostabiliser has detailed evidence of the whole of the test sequence; the latter is identical to the pre-flight check apart from the additional halts.

The installation check is initiated by operating the test button twice within 2 seconds. As in the pre-flight check, the white TESTING segment flashes when participation is required. If this is simply acknowledgement of the system status, the flashing will alternate with a flashing GO window (or flashing NO window in the case of three tests checking correct operation of the NO indication); the inspector restarts the sequence by pressing the test button.

## 11 TEST RIG

Prior the installation in the aircraft, the complete flight system is being evaluated on a ground test rig. This rig includes three engineers racks, each of which will house one computer and will contain an 8K core store module and other units required to monitor and control the computers. A central console housing the pilot's control unit, system sensors and junction box is also provided and an analogue computer is used for closed loop simulation work. An interface unit between one of the digital computers, the pilot's control unit and a general purpose digital computer is also provided to facilitate automatic processor fault injection work to be done.

## 12 SYSTEMS UNIT DETAILS

### 12.1 The digital computers

The computer incorporates a 16 bit parallel processor with a 5 bit function field and 11 bits of

direct memory addressing. It has two working registers and automatic sub-routine entry. Exist is achieved using a hardwired link pointer.

Some of the other main features are as follows:

Clock rate	8 MHz
Add/subtract time	3.75 us
Multiply/divide time	15 us
Programme store	3.5K PROM
Volatile store	0.75K BIPOLAR
A/D converter	12 bits, sample and hold time 100 $\mu$ s
D/A converter	10 bits
Input multiplexer	up to 32 inputs (of which 21 are sensor signals)
Output demultiplexer	5 outputs (of which 3 are control signals)
Update rate	55 ms

The input/output data and interlane data transmissions are carried out serially.

For the Sea King application the computer has been repackaged into a half ATR short case, fitted with a front end dog house for the power supplies. The unit contains 19 cards incorporating the input-output peripherals, interlane data transmission system and the engage-disconnect logic in addition to the central processor unit (CPU), the programme store and volatile store.

Whilst complete segregation between the three computer CPUs can be claimed, this is not possible for the peripherals involved in the interchange of data and discriminants. Careful consideration has consequently been given to the detailed design of the peripheral facilities in order to achieve good segregation. The dual in line TTL packages and other components are mounted on one side of the ten layer multilayer boards. Individual layers of the boards have been allocated for specific functions, including a number of exclusive zero and 5 volt supply planes. Minimum physical separation between individual tracks, connector pins and electronic packs associated with specific functions and redundancy of earth connections have also been specified. The cards are mounted in the computer case as dual modules with an ashlan segregation barrier between each module.

The rear face of each computer incorporates a 32 bit gearing pointer, the connections of which can be modified when the aircraft is grounded, to implement changes to a range of selected system parameters such as gearings, comparator settings etc. Allocation of 2 bits to one specific parameter allows for the selection of any one of four pre-defined values for this parameter.

A seven bar LRU indicator on the front of each computer facilitates the identification of faulty units located during pre-flight or in-flight testing. The indicator shows a single number between 1 and 9, thereby identifying a specific unit.

## 12.2 Pilot's control unit

The pilot's control unit has been designed to fit into the limited available space on the interseat console, replacing the autostabilisation controls of the normal AFCS. It was decided to incorporate both the pilot's control facilities and the engagement state and warning indicators in the one unit, although it is appreciated that for a production system separation of the control and warning functions might be operationally more desirable.

Poke-home contacts and flying leads have been used throughout the unit on order to facilitate easy maintenance and careful consideration has been given to lane segregation requirements within the unit.

The following facilities are provided:

- (a) Power supply switching to the three computers via the multi-pole power switch.
- (b) Engagement discriminant toggle pulses to each computer on pilot selection of ENGAGE via the POWER switch.
- (c) Red and amber system integrity warnings to the pilot.
- (d) A CHANNEL SELECT switch, allowing pilot selection of MULTIPLEX, SIMPLEX 1 or SIMPLEX 2 modes of system operation.
- (e) Three lane engagement state indicators showing lane IN, lane OUT and power OFF states.
- (f) On pilot selection, vertical gyro reject discriminants to the computers and indication of vertical gyro 1 and 2 states to the pilot.
- (g) On pilot selection, heading reject discriminants to the computers and indication of the heading engagement state to the pilot.
- (h) On pilot selection, ground test discriminant to the computers and indication of the testing state and result to the pilot.
- (i) Means of providing multiplex signals to the computers from pilot operated thumb wheels for pitch and roll trim.
- (j) Means for adding to the heading synchro signal a simplex heading trim signal derived from a

- (j) continued  
pilot operated knob.

### 12.3 Rate gyroscope unit

Identical triplex units are provided for all three axes using  $45^\circ$ /second rate gyroscopes, with a mounting frame suitable for installation to measure rates about any of the three specified axes. The three rate gyroscopes in each unit are aligned with their axes parallel in a machined mounting block which is fixed to the cast and machined mounting frame. A set of four fixing lugs are incorporated on two faces of the mounting frame and each set is accurately machined at right angles to the other. All three gyro units will be mounted on a single horizontal platform in the helicopter.

The rate gyroscope selected for the system is the Smiths Industries Ltd. Series 700 miniature gyro with an air bearing providing virtually zero bearing friction when running and consequently a very high reliability, with a predicted gyro MTBF of 20000 hours. A  $11^\circ$ /second maximum rate version of this gyro is currently being phased into service operation in the BEA Trident fleet as a replacement for ball bearing gyroscopes.

For the Sea King installation, demodulators provide DC output signals from the gyroscopes, and temperature compensation circuits are provided for both gain and datum characteristics of the gyros. These ensure a high degree of stability over the operating conditions and close inter-lane tracking of the gyro output signals.

Gimbal torquing facilities are provided in each rate gyroscope for use during the pre-flight tests to confirm the functionality and tracking accuracies of the three gyroscopes in each unit. The rotor speed of each rate gyro is monitored by means of an optical sensor system. This uses a fibre optic bundle, a light source, a photo transistor and associated electronics to generate output pulses at the gyro wheel speed. These pulses are transmitted to the digital computers where they are processed in order to detect a failure to achieve synchronous speed.

### 12.4 Stick position sensor units

Triplex cyclic stick position sensor units are used for both pitch and roll control. These are located in the aircraft control run such that identical units can be used for both axes. The output signals from each unit are obtained from three high precision plastic potentiometers, each of which is connected to the input lever through a clutch assembly, ensuring that the lever does not jam as a result of a seized potentiometer. The electrical and mechanical sections of the unit are segregated by a gear mounting plate and a high standard of electrical segregation has been maintained between the three potentiometers and associated wiring.

## 13 ASSESSMENT OF SYSTEM RELIABILITY AND INTEGRITY

13.1 Although the proving of software correctness was not referred to in the RAE's requirements, it is thought worthwhile to outline Smiths Industries approach to this matter; see section 14. The approach to assessment of the effect on the autostabiliser of defects in the system units and in associated equipment is discussed below. This failure analysis was restricted to the multiplex mode (except as required by the pre-flight check) by agreement with the RAE and its findings with regard to the system reliability and integrity, were as follows:-

- (i) Confirmation that any single defect of the system would have negligible effect upon the system performance.
- (ii) Confirmation that any multiple defects whose effect would not meet the integrity requirements in section 4.3, would have a total probability of less than  $10^{-7}$  per flying hour. In fact for each relevant fault sequence, regarded as a passive part followed by an active part, the probability of each of its parts is remote; i.e. less than  $10^{-10}$  per flying hour per sequence.
- (iii) An estimated MTBF substantially greater than  $10^5$  flying hours for the initial comparator settings; each setting is 2 deg per sec. which accommodates all analogue tolerances.

13.2 The failure analysis philosophy:- It is essential that a failure assessment, i.e. the record of a failure analysis, conveys confidence in the completeness of the analysis. In Smiths Industries experience this necessitates both bottom-up and top-down procedures except for simple systems without ICs. Even in the latter case, the use of both types of procedure could well be more efficient than a bottom-up procedure alone. System reliability and integrity requirements are expressible qualitatively or quantitatively, i.e. in terms of fault-tolerance or probability of effects. However, the approach to failure analysis is essentially the same, so it is unimportant that for a given system both modes of expression are likely to be involved.

Considering firstly bottom-up procedures; for reliability and integrity requirements in terms of fault tolerance, such a procedure nominally relies on knowledge of every component (including interconnection) failure mode that could conceivably occur in service. However, for proprietary IC components such knowledge is normally, and is likely to remain, unavailable; the IC manufacturer being naturally more interested in failure mechanisms. Further, the in-service returns for complex ICs cannot be relied on for guidance as to conceivably-occurring failure modes. For example, suppose the true classification of a particular IC failure is 'Wrong state table' but that diagnosis at module and component levels found, quite adequately for the particular application, the IC as 'Output stuck at 1'. Then the latter is likely to be the in-service return. When the requirements are in terms of probability of effects it is possible that a particular classification would have an insignificant failure rate and thus could be ignored, otherwise the situation is no clearer than for requirements in terms of fault tolerance. Thus a bottom-up procedure, such as the usual way in which an FMEA is conducted, has to rely to a certain extent in practice on the analyst



postulating component failure modes that could conceivably occur in service.

Considering now top-down procedures; for reliability and integrity requirements in terms of fault tolerance, such a procedure relies on the analyst considering every locality failure mode that is both possibly relevant to the requirements and of conceivable occurrence in service and postulating a sufficient set for the failure analysis. Provided the redundancy is on a regional rather than a component basis and is in general implemented with matching segregation, such a task has been found practicable. That is, that (i) the analyst can satisfy himself that the postulated locality failure modes are a sufficient set and (ii) the failure assessment can convey the analyst's confidence e.g. to equipment certification authorities, principally by showing the incompatibility of these failure modes whose occurrence in service is conceivable but are not postulated with failure modes that could possibly be relevant to the requirements. When the requirements are in terms of probability of effects, a top-down procedure relies on the analyst considering every locality failure mode that is possibly relevant to the requirements and would have a significant rate of occurrence in service. Thus the task of postulating a "sufficient set" is somewhat eased.

### 13.3 Application of the failure analysis philosophy

Because redundancy in this system is wholly on a regional basis, and has been implemented with complete segregation (i.e. bridging between redundant regions is inconceivable) or with good segregation (i.e. the occurrence in service of bridging between redundant regions is inconceivable) with only minor exceptions, a top-down approach to the whole failure analysis was selected as being the most efficient. Two examples of locality failure modes that were postulated are Spurious Bit and Open-circuit Interconnection. Because the latter is also a component failure mode there were no sub-classes but the former needed three, viz. Stuck at 0, Stuck at 1 and Soft Output. Also sub-sub-classes were necessary for Soft Output, to cover the three types of response that could conceivably occur at a fan-out:-

Soft Output (0), i.e. at least one load sees the driver as Stuck at 0 and the remainder see it behaving normally.

or Soft Output (1), i.e. at least one load sees the driver as Stuck at 1 and the remainder see it behaving normally.

or Soft Output (0 & 1), i.e. all loads see the driver as Stuck at 0 or 1 and at least one cognisance differs from the remainder.

Faulty Processor was of course one of the postulated locality failure modes and needed three sub-classes, viz. (a) Program unaffected, (b) Slight Departure from Normal Program (i.e. the changed program length does not desynchronise), and (c) Gross Departure from Normal Program (i.e. synchronisation is lost). Also (b) and (c) needed three and four sub-sub-classes respectively, giving the following eight types of defect:-

A type (a) defect involves one location in the data store being affected; thus several data words may be spurious.

A type (b1) defect involves non-execution of one INP instruction.

A type (b2) defect involves non-execution of one OUT instruction.

A type (b3) defect involves one OUT instruction being preceded by a spurious sequence.

A type (c1) defect involves two or more OUT instructions, each one having a type (iii) or a type (iv) defect.

A type (c2) defect involves repetition of one instruction.

A type (c3) defect involves processor stopped.

A type (c4) defect involves processor running wild.

## 14 SOFTWARE CORRECTNESS

### 14.1 Software design

In a computer context the word software is used to describe that part of the system which is particular to the application and which directs the otherwise general hardware to fulfil the requirements of the application. The term may be defined more generally and more usefully as: Software is that which structures a set of functional elements to fulfil a task requirement. This latter definition allows the concept of levels of software. Any system looked at at any particular level of details may be considered as a set of functional elements being caused to interact in a manner to fulfil a task requirement. Thus at one level the functional elements may be individual computer instructions and the task may be to provide the function of, for example, a limit. At another level the functional elements may be functions providing integration, first order lag, gain schedule limit etc. and the task may be to provide a particular control law. This concept of levels of software is crucial to a software design procedure appropriate to the production of correct software.

The software design procedure adopted is that which may reasonably be called structured and which takes place in a top down manner. The essential feature of structured programming is the appreciation of levels of program and the need at each level to define precisely the functions of the software modules and the interfaces between them. The highest level of software is the statement of requirements for the system. This statement defines the way in which the system is required to perform in the environment of aircraft characteristics, aircraft motions, pilot inputs and hardware failures.



The first task in the production of a structured program is to define a small but convenient number of functional sub-divisions of the total task and to specify precisely the interfaces between them.

At the second level each of these functional sub-divisions is itself defined in terms of a small number of sub-functions with precisely specified interfacing.

Progressing in this way the tasks can be broken down into a series of levels. At the lowest level the basic functional elements are the machine instructions of the CPU and the functions generated are the simplest sub-routines.

Looking then at the structure which leads in steps from the system requirement to the machine instructions, at each step functions are defined as combinations of simpler functions similarly defined at a lower stage. Thus at each stage the basic elements are the functions which have been defined at the next lower stage and the software concerns the grouping of these basic functional elements to perform a more complex function. The task of software proving is therefore at each stage to analyse the assertion that one particular combination of elements, having defined properties, constitutes one particular function and none other. Thus step by step one can proceed from the behaviour of the system to the basic characteristics of the computing elements.

A typical series of steps is:

1. The system requirement can be accomplished by performing a given series of system tasks.
2. Each system task can be accomplished by performing a particular series of tasks.
3. Each task can be accomplished by performing a particular series of sub-tasks.
4. Each sub-task can be accomplished by performing a particular series of sub-routines.
5. Each sub-routine can be accomplished by performing a particular series of machine instructions.

#### 14.2 Software design analysis

The task of software design analysis is to establish that the system task requirement is achieved by the sequence of actions called for by the software. The structured manner of the software design allows a similarly structured approach to the analysis of the design.

At each stage in the sequence which links the properties of the basic computing elements to the overall system task, a set of functions is defined, each of which, it is asserted, can be represented by logical combinations of certain simpler functions. The task of software proving is to examine the validity of these logical assertions. In principle, the method is the same at all levels. The total logical consequences of the proposed combination of simpler functions is written down and equated to the logic of the claimed function. If they are identical the assertion is proved.

The difficulty of the task of proving depends very much on the characteristics of the elements; being most difficult in those areas in which a concise definition of the task is lacking, e.g. at the higher software levels. In such difficult areas the method of verification tends to be by simulation and test rather than by a more formal paper analysis.

#### 14.3 Relevance of the level of computer language

As in the software design process successively lower and more detailed levels are reached it becomes necessary to consider the level of language in which the task is to be expressed to the digital processor. The principal alternatives are:-

1. Assembler code in which these mnemonics, or functional symbol, used have a 1 : 1 relationship to the machine code on which the processor operates.
2. A high level language in which a computer takes on the task of converting functional statements into sequences of machine code to cause the desired functions to be performed.

The latter has an immediate appeal in that more 'English like' statements can be written, giving the software an appearance of visibility. This characteristic of visibility is achieved by delegating the detailed processor operation to be organised by the compiler in response to the high level of statements. However, this means that the compiler itself is an integral part in the chain which produces the machine instructions. In which case, either the validity of the compiler itself must be proved to a level appropriate to something which can have a coherent effect on the multilane system or the machine code which it generates must be analysed to ensure that the required functions are being correctly performed. The former is a task of extreme difficulty and the need for the latter removes the apparent advantage of the high level language.

The use of assembler level programming in conjunction with a structured programming approach does not possess these disadvantages. The tasks which are to be coded are the lowest level of sub-routine performing simple functions such as Limit, First order lag, Check for identity etc. for which an assembler language program is appropriate and since the assembler provides 1 : 1 relationship between the symbolic input and the single line of code the problem of establishing its validity by direct test is simple. With regard to visibility, because the total task is structured in such a way that each assembler routine has a small well-defined task to perform and that larger functions are achieved by grouping sequences of these simpler tasks, then the visibility is at least as good as with a higher level language.

## 15 FUTURE SYSTEMS IMPROVEMENTS

The need to provide redundancy for outer loop control facilities will depend upon the type of mission being undertaken. However, further developments of the system described herein to incorporate the autopilot facilities normally available in a modern helicopter and facilities for coupling to modern guidance aids can be achieved by computer programme extension and sensor input implementation.

Modern helicopters are now being designed with semi-rigid rotors which simplify the hub design but introduce additional high speed stability problems. Future developments in the helicopter design are likely to be towards rigid rotors, for which the material problems are understood to be less severe. However, such designs will introduce additional speed dependent cross coupling effects, and to counteract these a versatile automatic control system will be required. It is considered that this is more likely to be realisable using digital rather than analogue techniques.

Rapid developments of digital devices suitable for airborne application are now taking place; for example the LSI processor mentioned in section 5, which occupies two boards, has a greater capability than the SDC10, which occupies 6 out of 19 boards. Also under development at Smiths Industries is the utilisation of both microprocessor slices and 'complete' microprocessors. Similar developments are taking place in peripheral devices such as A/D and D/A converters, multiplexers, etc.

Development of fibre optic data transmission systems and multi-sensor digital data acquisition systems with good RF noise rejection capabilities have also been carried out by Smiths Industries, with a view to integration into the next generation of flight control systems.

## 16 CONCLUSIONS

This paper highlights pilot work load as a major obstacle to the tactical use of helicopters at night and in poor visibility. The work load can be reduced by the fitting of an autostabiliser to the aircraft but the maximum gain will not be realised unless there is adequate pilot confidence in the integrity of the system. A defect-survival system can give this confidence and such a system has been selected for flight evaluation in a helicopter at the Royal Aircraft Establishment. Further developments in the area of digital, multiplex flight control system depend largely on future operational requirements and aircraft design trends.

## REFERENCES

1. J.F.O. Evans, K.A. Helps: An experimental investigation into duplex digital control of an engine with reheat. AGARD Conference, Geilo, Norway, Paper No. 16, September 1973.
2. D. Kimberley, P.W.J. Fullam: Flight control system development in the UK. RAE Technical Memorandum Avionics 151, presented at AGARD Conference, Geilo, Norway, September 1973.
3. H.B. Johnson: The effects of semi-rigid rotors on helicopter autostabiliser design. AGARD Conference, Konstanz, Paper No. 14 June 1971.

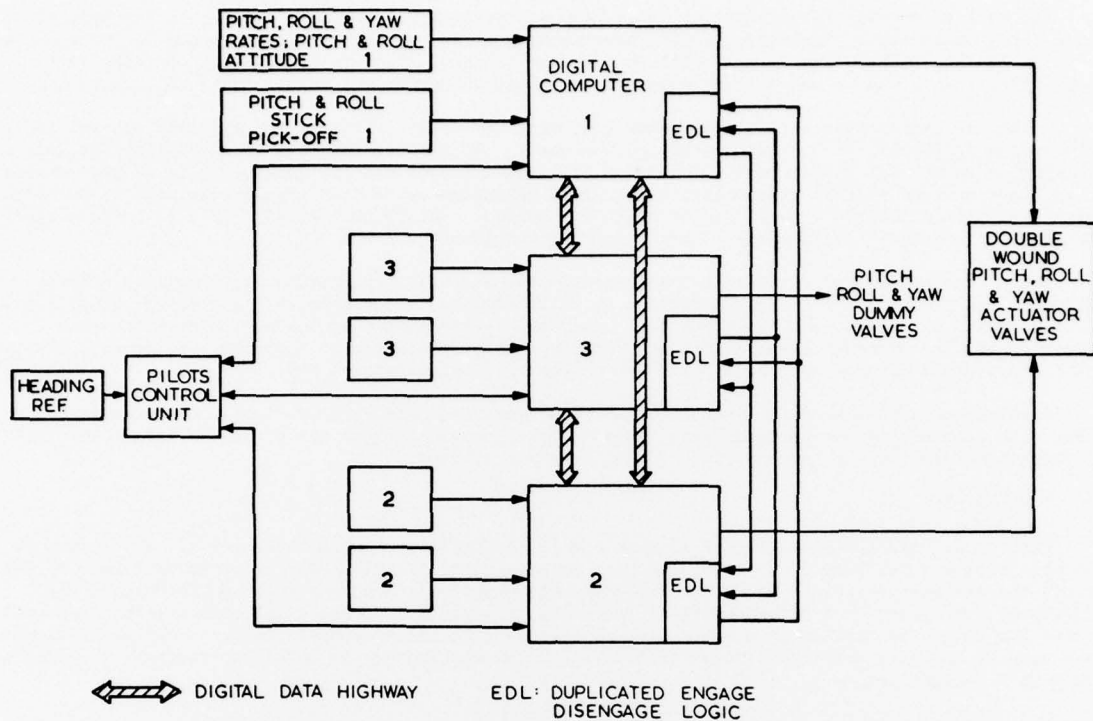


Fig.1 Schematic of triplex configuration

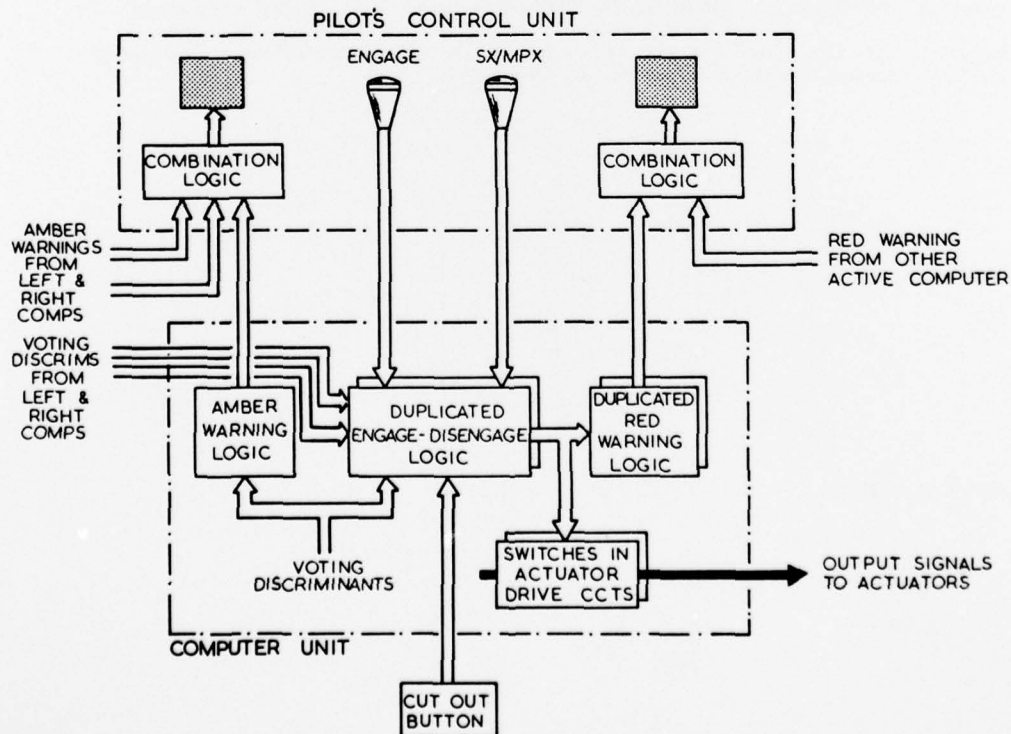


Fig.2 Schematic diagram of engage/disengage and warning logic

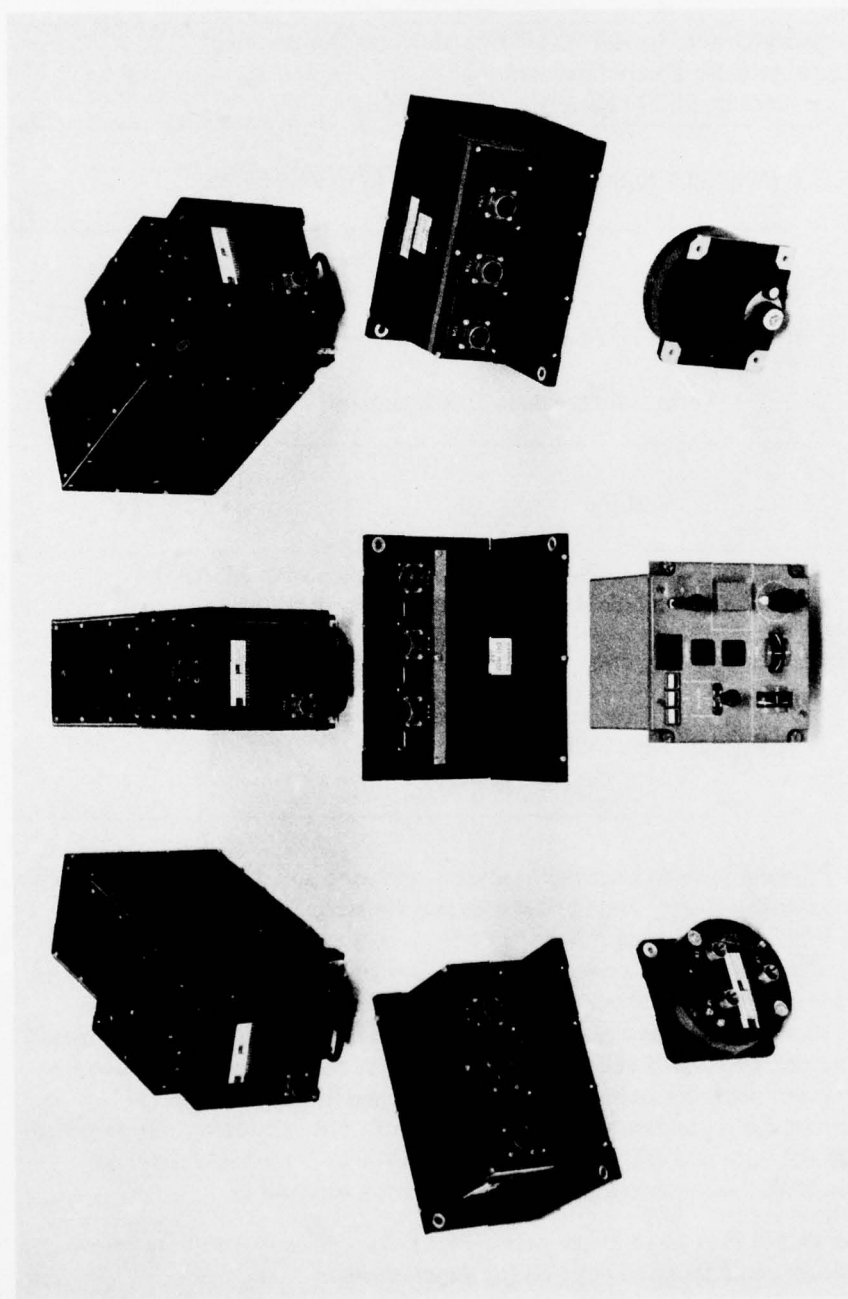


Fig. 3 The units of the Sea King defect survival autostabiliser system

Back Row: Digital computers  
 Centre Row: Rate gyro units  
 Front Row: Stick position sensor unit, pilots control unit, stick position sensor unit



24-16

REPORT DOCUMENTATION PAGE			
1. Recipient's Reference	2. Originator's Reference AGARD-AG-224	3. Further Reference ISBN 92-835-0192-6	4. Security Classification of Document UNCLASSIFIED
5. Originator	Advisory Group for Aerospace Research and Development North Atlantic Treaty Organization 7 rue Ancelle, 92200 Neuilly sur Seine, France		
6. Title	INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS		
7. Presented at			
8. Author(s) Various	Technical Director: P.R.Kurzahls		9. Date April 1977
10. Author's Address Various			11. Pages 384
12. Distribution Statement	This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.		
13. Keywords/Descriptors	<div style="display: flex; justify-content: space-between;"> <div> Flight control Avionics Reliability (electronics) </div> <div> Design criteria Airborne equipment Man machine systems </div> <div> Solid state devices </div> </div>		
15. Abstract	<p>The intent of the AGARDograph is to address the hardware, software and man-machine interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some increases in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.</p> <p>The AGARDograph is organized into three major parts. Part I, <i>Background and Requirements</i>; Part II, <i>Analysis and Testing</i>; and Part III, <i>Design and Implementation</i>.</p> <p>This AGARDograph was prepared at the request of the Guidance and Control Panel of AGARD-NATO.</p>		

24-16

<p>AGARDograph No.224 Advisory Group for Aerospace Research and Development, NATO INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS Technical Director: P.R.Kurzahls Published April 1977 384 pages</p> <p>The intent of the AGARDograph is to address the hardware, software and man-machine interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on</p> <p>P.T.O.</p>	<p>AGARD-AG-224</p> <p>Flight control Avionics Reliability (electronics) Design criteria Airborne equipment Man machine systems Solid state devices</p>	<p>AGARDograph No.224 Advisory Group for Aerospace Research and Development, NATO INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS Technical Director: P.R.Kurzahls Published April 1977 384 pages</p> <p>The intent of the AGARDograph is to address the hardware, software and man-machine interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on</p> <p>P.T.O.</p>	<p>AGARD-AG-224</p> <p>Flight control Avionics Reliability (electronics) Design criteria Airborne equipment Man machine systems Solid state devices</p>
<p>AGARDograph No.224 Advisory Group for Aerospace Research and Development, NATO INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS Technical Director: P.R.Kurzahls Published April 1977 384 pages</p> <p>The intent of the AGARDograph is to address the hardware, software and man-machine interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on</p> <p>P.T.O.</p>	<p>AGARD-AG-224</p> <p>Flight control Avionics Reliability (electronics) Design criteria Airborne equipment Man machine systems Solid state devices</p>	<p>AGARDograph No.224 Advisory Group for Aerospace Research and Development, NATO INTEGRITY IN ELECTRONIC FLIGHT CONTROL SYSTEMS Technical Director: P.R.Kurzahls Published April 1977 384 pages</p> <p>The intent of the AGARDograph is to address the hardware, software and man-machine interface aspects of reliable flight control systems. Rapid advances in solid-state electronics which resulted in a hundred-fold decrease in computer size, power and cost over the past two decades have revolutionized the design of modern flight control systems. Designers have capitalized on</p> <p>P.T.O.</p>	<p>AGARD-AG-224</p> <p>Flight control Avionics Reliability (electronics) Design criteria Airborne equipment Man machine systems Solid state devices</p>

<p>these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some increases in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.</p> <p>The AGARDograph is organized into three major parts. Part I, <i>Background and Requirements</i>; Part II, <i>Analysis and Testing</i>; and Part III, <i>Design and Implementation</i>.</p> <p>This AGARDograph was prepared at the request of the Guidance and Control Panel of AGARD-NATO.</p> <p>ISBN 92-835-0192-6</p>	<p>these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some increases in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.</p> <p>The AGARDograph is organized into three major parts. Part I, <i>Background and Requirements</i>; Part II, <i>Analysis and Testing</i>; and Part III, <i>Design and Implementation</i>.</p> <p>This AGARDograph was prepared at the request of the Guidance and Control Panel of AGARD-NATO.</p> <p>ISBN 92-835-0192-6</p>
<p>these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some increases in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.</p> <p>The AGARDograph is organized into three major parts. Part I, <i>Background and Requirements</i>; Part II, <i>Analysis and Testing</i>; and Part III, <i>Design and Implementation</i>.</p> <p>This AGARDograph was prepared at the request of the Guidance and Control Panel of AGARD-NATO.</p> <p>ISBN 92-835-0192-6</p>	<p>these gains primarily by incorporating additional control functions to improve aircraft or weapon system performance and survivability. As a result, control system complexity also has increased by 1 to 2 orders of magnitude, and highly-reliable flight control system operation has become critically important to mission planning and execution. While some increases in system reliability were obtained through redundancy in system mechanization, concerted efforts aimed at improving system integrity were not initiated until the late 1960's. This AGARDograph summarizes associated analysis, design, development and checkout approaches.</p> <p>The AGARDograph is organized into three major parts. Part I, <i>Background and Requirements</i>; Part II, <i>Analysis and Testing</i>; and Part III, <i>Design and Implementation</i>.</p> <p>This AGARDograph was prepared at the request of the Guidance and Control Panel of AGARD-NATO.</p> <p>ISBN 92-835-0192-6</p>



AGARD

NATO  OTAN

7 RUE ANCELLE · 92200 NEUILLY-SUR-SEINE  
FRANCE

Telephone 745.08.10 · Telex 610176

DISTRIBUTION OF UNCLASSIFIED  
AGARD PUBLICATIONS

AGARD does NOT hold stocks of AGARD publications at the above address for general distribution. Initial distribution of AGARD publications is made to AGARD Member Nations through the following National Distribution Centres. Further copies are sometimes available from these Centres, but if not may be purchased in Microfiche or Photocopy form from the Purchase Agencies listed below.

NATIONAL DISTRIBUTION CENTRES

**BELGIUM**

Coordonnateur AGARD – VSL  
Etat-Major de la Force Aérienne  
Caserne Prince Baudouin  
Place Dailly, 1030 Bruxelles

**CANADA**

Defence Scientific Information Service  
Department of National Defence  
Ottawa, Ontario K1A 0Z2

**DENMARK**

Danish Defence Research Board  
Østerbrogades Kaserne  
Copenhagen Ø

**FRANCE**

O.N.E.R.A. (Direction)  
29 Avenue de la Division Leclerc  
92 Châtillon sous Bagneux

**GERMANY**

Zentralstelle für Luft- und Raumfahrt-  
dokumentation und -information  
Postfach 860880  
D-8 München 86

**GREECE**

Hellenic Armed Forces Command  
D Branch, Athens

**ICELAND**

Director of Aviation  
c/o Flugrad  
Reykjavik

**ITALY**

Aeronautica Militare  
Ufficio del Delegato Nazionale all'AGARD  
3, Piazzale Adenauer  
Roma/EUR

**LUXEMBOURG**

See Belgium

**NETHERLANDS**

Netherlands Delegation to AGARD  
National Aerospace Laboratory, NLR  
P.O. Box 126  
Delft

**NORWAY**

Norwegian Defence Research Establishment  
Main Library  
P.O. Box 25  
N-2007 Kjeller

**PORTUGAL**

Direcção do Serviço de Material  
da Força Aérea  
Rua da Escola Politécnica 42  
Lisboa  
Attn: AGARD National Delegate

**TURKEY**

Department of Research and Development (ARGE)  
Ministry of National Defence, Ankara

**UNITED KINGDOM**

Defence Research Information Centre  
Station Square House  
St. Mary Cray  
Orpington, Kent BR5 3RE

**UNITED STATES**

National Aeronautics and Space Administration (NASA),  
Langley Field, Virginia 23365  
Attn: Report Distribution and Storage Unit

THE UNITED STATES NATIONAL DISTRIBUTION CENTRE (NASA) DOES NOT HOLD  
STOCKS OF AGARD PUBLICATIONS, AND APPLICATIONS FOR COPIES SHOULD BE MADE  
DIRECT TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS) AT THE ADDRESS BELOW.

PURCHASE AGENCIES

*Microfiche or Photocopy*

National Technical  
Information Service (NTIS)  
5285 Port Royal Road  
Springfield  
Virginia 22151, USA

*Microfiche*

Space Documentation Service  
European Space Agency  
10, rue Mario Nikis  
75015 Paris, France

*Microfiche*

Technology Reports  
Centre (DTI)  
Station Square House  
St. Mary Cray  
Orpington, Kent BR5 3RF  
England

Requests for microfiche or photocopies of AGARD documents should include the AGARD serial number, title, author or editor, and publication date. Requests to NTIS should include the NASA accession report number. Full bibliographical references and abstracts of AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR),  
published by NASA Scientific and Technical  
Information Facility  
Post Office Box 8757  
Baltimore/Washington International Airport  
Maryland 21240, USA

Government Reports Announcements (GRA),  
published by the National Technical  
Information Services, Springfield  
Virginia 22151, USA



Printed by Technical Editing and Reproduction Ltd  
Harford House, 7-9 Charlotte St, London W1P 1HD

ISBN 92-835-0192-6